
COMBINING APPEARANCE, DEPTH AND MOTION FOR EFFICIENT SEMANTIC SCENE UNDERSTANDING

Dissertation approved by the
Fachbereich Informatik

in fulfillment of the requirements for the degree of
Doktor-Ingenieur (Dr.-Ing.)

by

TIMO REHFELD

Diplom Ingenieur (Dipl.-Ing.)
born in Cologne, Germany

Examiner: Prof. Stefan Roth, PhD

Co-examiner: Prof. Carsten Rother, PhD

Darmstadt, 2017

Timo Rehfeld
Combining Appearance, Depth and Motion for Efficient Semantic
Scene Understanding

Darmstadt, Technische Universität Darmstadt
Year of Publication on TUprints: 2018
Date of Defense: September 26, 2017

Published under CC BY-NC 4.0 International
<https://creativecommons.org/licenses/>

ABSTRACT

Computer vision plays a central role in autonomous vehicle technology, because cameras are comparably cheap and capture rich information about the environment. In particular, object classes, *i.e.* whether a certain object is a pedestrian, cyclist or vehicle can be extracted very well based on image data. Environment perception in urban city centers is a highly challenging computer vision problem, as the environment is very complex and cluttered: road boundaries and markings, traffic signs and lights and many different kinds of objects that can mutually occlude each other need to be detected in real-time. Existing automotive vision systems do not easily scale to these requirements, because every problem or object class is treated independently. Scene labeling on the other hand, which assigns object class information to every pixel in the image, is the most promising approach to avoid this overhead by sharing extracted features across multiple classes. Compared to bounding box detectors, scene labeling additionally provides richer and denser information about the environment. However, most existing scene labeling methods require a large amount of computational resources, which makes them infeasible for real-time in-vehicle applications. In addition, in terms of bandwidth, a dense pixel-level representation is not ideal to transmit the perceived environment to other modules of an autonomous vehicle, such as localization or path planning.

This dissertation addresses the scene labeling problem in an automotive context by constructing a scene labeling concept around the “Stixel World” model of Pfeiffer (2011), which compresses dense information about the environment into a set of small “sticks” that stand upright, perpendicular to the ground plane. This work provides the first extension of the existing Stixel formulation that takes into account learned dense pixel-level appearance features. In a second step, Stixels are used as primitive scene elements to build a highly efficient region-level labeling scheme. The last part of this dissertation finally proposes a model that combines both pixel-level and region-level scene labeling into a single model that yields state-of-the-art or better labeling accuracy and can be executed in real-time with typical camera refresh rates. This work further investigates how existing depth information, *i.e.* from a stereo camera, can help to improve labeling accuracy and reduce runtime.

ZUSAMMENFASSUNG

Maschinelle Bildverarbeitung spielt eine zentrale Rolle für autonome Fahrzeuge, da Kameras vergleichsweise günstig sind und eine Vielzahl an Informationen über die Umgebung erfassen. Insbesondere die Objektklasse, also ob ein bestimmtes Objekt ein Fußgänger, Radfahrer oder Auto ist, kann sehr gut anhand von Bildmaterial erkannt werden. Umgebungserfassung im städtischen Umfeld ist eine große Herausforderung für Bildverarbeitungsalgorithmen, da die Umgebung sehr komplex und unstrukturiert ist: Fahrbahnberandung und Spurmarkierungen, Schilder und Ampeln, und viele weitere Objekte die sich gegenseitig verdecken können, müssen in Echtzeit erkannt werden. Die derzeit existierenden Algorithmen in intelligenten Fahrzeugen skalieren nicht ohne weiteres zu diesen Anforderungen, da jedes Problem bzw. jede Objektklasse getrennt behandelt wird. Sogenanntes "Scene Labeling", welches jedem Pixel im Bild eine Klasse zuweist, ist eine vielversprechende Methode um diesen Mehraufwand zu vermeiden indem extrahierte Bildmerkmale zwischen verschiedenen Klassen geteilt werden. Verglichen mit Bounding-Box Detektoren liefert Scene Labeling außerdem eine reichhaltigere und dichtere Darstellung der Umgebung. Die meisten bestehenden Scene Labeling Verfahren haben allerdings einen sehr hohen Rechenaufwand, was eine Anwendung in Echtzeit nicht ermöglicht. Zusätzlich ist im Hinblick auf Bandbreite eine dichte Darstellung auf Pixel-Ebene nicht ideal um die erfasste Umgebung an andere Module in einem autonomen Fahrzeug (wie z.B. Lokalisierung und Pfad-Planung) zu übertragen.

Diese Dissertation geht Scene Labeling aus einem Automobil-Kontext an, indem ein Scene Labeling Konzept um das "Stixel Welt" Modell von Pfeiffer (2011) aufgebaut wird, welches die dichte Umgebungsinformation zu einer Menge von kleinen senkrecht auf dem Boden stehenden Stäben komprimiert. In dieser Arbeit wird erstmals die bestehende Stixel-Formulierung dahingehend erweitert, dass dichte gelernte Bildmerkmale auf Pixel-Ebene berücksichtigt werden. In einem zweiten Schritt werden Stixel als Basisbausteine der Szene benutzt um ein hocheffizientes Labeling Schema auf Regions-Ebene zu realisieren. Der letzte Teil dieser Arbeit stellt ein Konzept vor, dass Labeling auf Pixel-Ebene und Regions-Ebene in einem einzigen Modell kombiniert, welches eine Genauigkeit vergleichbar oder besser als der aktuelle Stand der Technik liefert und in Echtzeit mit typischen Bildwiederholraten ausgeführt werden kann. Diese Arbeit untersucht des Weiteren inwiefern vorhandene Tiefeninformation, z.B. von einer Stereo-Kamera, helfen kann um die Labeling-Präzision zu erhöhen und Laufzeit zu reduzieren.

PUBLICATIONS

Timo Scharwächter¹, Markus Enzweiler, Uwe Franke, Stefan Roth, Efficient Multi-Cue Scene Segmentation, *German Conference on Pattern Recognition (GCPR) [Best Paper Award]*, 2013

Timo Scharwächter¹, Markus Enzweiler, Uwe Franke, Stefan Roth, Stixmantics: A Medium-Level Model for Real-Time Semantic Scene Understanding, *European Conference on Computer Vision (ECCV)*, 2014

Timo Scharwächter¹, Manuela Schuler, Uwe Franke, Visual Guard Rail Detection for Advanced Highway Assistance Systems *IEEE Intelligent Vehicles Symposium (IV)*, 2014

Timo Scharwächter¹, Uwe Franke, Low-Level Fusion of Color, Texture and Depth for Robust Road Scene Understanding, *IEEE Intelligent Vehicles Symposium (IV)*, 2015

Marius Cordts, Mohamed Omran, Sebastian Ramos, **Timo Scharwächter**¹, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele, The Cityscapes Dataset, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR): Workshop on The Future of Datasets in Vision*, 2015

Marius Cordts, Mohamed Omran, Sebastian Ramos, **Timo Rehfeld**, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele, The Cityscapes Dataset for Semantic Urban Scene Understanding, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016

Marius Cordts², **Timo Rehfeld**², Markus Enzweiler, Uwe Franke, Stefan Roth, Tree-Structured Models for Efficient Multi-Cue Scene Labeling, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017

Marius Cordts², **Timo Rehfeld**², Lukas Schneider², David Pfeiffer, Markus Enzweiler, Stefan Roth, Marc Pollefeys, Uwe Franke, The Stixel World: A Medium-Level Representation of Traffic Scenes, *Special Issue in Image & Vision Computing (IVC), Automotive Vision: Challenges, Trends, Technologies and Systems for Vision-Based Intelligent Vehicles*, 2017

¹ Last name changed from Scharwächter to Rehfeld in 2015

² Authors contributed equally

ACKNOWLEDGMENTS

I would like to thank my supervisor Stefan Roth, for agreeing to supervise me as an external Ph.D. student, which allowed me to gain experience in academic research and to work at a company at the same time. However, more importantly, thank you for always providing high quality technical feedback and for your commitment close to paper deadlines.

I also want to express my warmest gratitude to Uwe Franke, who always put trust in me but at the same time gave me many impulses for new ideas and always pushed me towards finding a more elegant solution. You have been a mentor for me in many ways. Thank you also for shielding my research from the distractions of a corporate environment. I know now that this cannot be taken for granted.

Thanks to all members of the Image Understanding Group at Daimler for creating such a great, challenging, and fun place to work, which ultimately was one of the main reasons I decided to join a Ph.D. program in the first place. I found many long-term friends in this group. Special thanks to David Pfeiffer, Markus Enzweiler, and Marius Cordts for the close collaboration. This dissertation would not have been possible without all our in-depth discussions, joint code development and paper writing.

I would like to thank my parents for always supporting me and giving me the freedom to set and follow my own goals. Thanks to my friends in Cologne, for always having an open door and giving me the feeling that I was never really gone. This safe harbor really means a lot to me. And after all, my biggest gratitude goes to my wife Kira for her support, continuous trust and patience during all the weekends I spent writing this dissertation.

CONTENTS

1	INTRODUCTION	1
1.1	Motivation	2
1.1.1	Automotive Scene Labeling	2
1.1.2	Representing the Visual Environment	3
1.1.3	Combining Multiple Cues	4
1.2	Problem Statement	5
1.2.1	Why Urban Scenes?	5
1.2.2	Industrial Considerations	6
1.2.3	Approach	7
1.3	Contributions	8
1.4	Outline	9
2	BACKGROUND AND RELATED WORK	11
2.1	Reasoning from Motion and Stereo	11
2.2	The Stixel World	14
2.2.1	Graphical Model	15
2.2.2	Inference	19
2.3	The Bag-of-Features Model	19
2.3.1	Feature Encoding Methods	20
2.3.2	Pyramidal Encoding	21
2.4	Scene Labeling	22
2.4.1	Pixel- and Region-level Scene Labeling	23
2.4.2	Temporally Consistent Scene Labeling	24
2.4.3	Scene Labeling Reference Methods	25
3	DATASETS AND METRICS	31
3.1	Datasets for Outdoor Urban Scene Labeling	32
3.1.1	CamVid	32
3.1.2	Leuven	33
3.1.3	KITTI	33
3.1.4	Discussion	34
3.2	The Daimler Urban Segmentation Dataset	35
3.2.1	General Description	35
3.2.2	Evaluation Protocol	38
3.2.3	Dataset Statistics	41
3.3	The Static Estimator	44
I	PIXEL LEVEL	49
4	SCENE LABELING AS CLASSIFICATION OF LOCAL IMAGE PATCHES	51
4.1	Limitations of the Stixel Model	52
4.2	Local Patch Classification	53
4.2.1	Feature Types	54
4.2.2	Decision Forests for Patch Classification	59

4.3	Extending the Stixel World	62
4.3.1	Updated Graphical Model	63
4.4	Experiments	65
4.4.1	RDF Parameters	65
4.4.2	Feature Type Comparison	66
4.4.3	Runtime	68
4.4.4	Stixel Extension	69
4.4.5	Qualitative Results	74
4.4.6	Comparison to Reference Methods	76
4.5	Discussion	79
II	REGION LEVEL	81
5	SCENE LABELING AS CLASSIFICATION OF PROPOSAL RE- GIONS	83
5.1	Generation and Classification of Proposal Regions . . .	84
5.1.1	Efficient Stixel-based Region Generation	85
5.1.2	Decision Forests for Region Encoding	89
5.1.3	Benefits of Depth for Encoding	91
5.1.4	Region Classification	93
5.2	Spatio-Temporal Regularization	94
5.2.1	Model Considerations	94
5.2.2	Temporal Integration	96
5.2.3	Spatial CRF Model	98
5.3	Experiments	100
5.3.1	Region Encoding	100
5.3.2	Stixel-based Regions	102
5.3.3	Spatio-temporal Regularization	105
5.3.4	Runtime	106
5.3.5	Comparison to Reference Methods	108
5.4	Discussion	109
III	COMBINING PIXEL AND REGION LEVEL	111
6	A JOINT MODEL FOR EFFICIENT ENCODING AND CLAS- SIFICATION	113
6.1	Encode-and-Classify Trees	114
6.1.1	Conflicting Demands of Encoding and Classifi- cation	116
6.1.2	Sub-trees for Feature Encoding	117
6.1.3	Range Restriction	118
6.1.4	Training details	118
6.2	Final Concept	120
6.3	Experiments	120
6.3.1	Conflicting Parameters	120
6.3.2	Comparison to Reference Methods	122
6.3.3	Runtime	124
7	DISCUSSION AND OUTLOOK	127
7.1	Discussion and Limitations of the Approach	127

7.2	Future Perspectives	130
7.2.1	Segmentation Trees	131
7.2.2	Scene Labeling Datasets	131
7.2.3	Deep Learning	132
7.3	Final Thoughts	132
BIBLIOGRAPHY		135

LIST OF FIGURES

Figure 1.1	Scene labeling example	2
Figure 2.1	Examples of the 6D-Vision approach	12
Figure 2.2	Dense stereo depth map and Stixels	13
Figure 2.3	Motion segmentation of Stixels	13
Figure 2.4	Stixel world as a factor graph	15
Figure 2.5	Concept of ERC forests	21
Figure 2.6	Concept of spatial pyramid matching	22
Figure 2.7	Overview of the ALE labeling method	26
Figure 2.8	Overview of the RCPN labeling method	27
Figure 2.9	Overview of the Layered Interpret. method	28
Figure 2.10	Overview of the Semantic Texton method	29
Figure 2.11	Bag of semantic textons	29
Figure 3.1	Different scene labeling datasets	34
Figure 3.2	Bertha Benz Memorial Route	36
Figure 3.3	Examples of our DUS dataset	37
Figure 3.4	Artificial confusion matrix with three common measures for semantic segmentation	41
Figure 3.5	Pixel-level spatial location priors for the classes in our <i>DUS</i> dataset	43
Figure 3.6	Class-specific distributions of 3D points in our dataset	45
Figure 3.7	Best class label based on spatial location	46
Figure 4.1	Typical problems of Stixels	53
Figure 4.2	Coarse pixel-level scene labeling result	54
Figure 4.3	Filter kernels used for texture encoding	56
Figure 4.4	Visualization of different feature channels	56
Figure 4.5	Visualization of our camera model	58
Figure 4.6	Decision trees for pixel classification	61
Figure 4.7	Probability maps for different classes	63
Figure 4.8	Extended Stixel world as a factor graph	64
Figure 4.9	Evaluation of central RDF parameters	67
Figure 4.10	Pixel classifier runtime	72
Figure 4.11	Qualitative results of the pixel classifier and its integration into Stixels (<i>DUS</i>)	77
Figure 4.12	Qualitative results of the pixel classifier and its integration into Stixels (<i>KITTI</i>)	78
Figure 5.1	Region-level concept overview	85
Figure 5.2	Artificial region generation example	86
Figure 5.3	Stixel-based region generation results	88
Figure 5.4	Decision trees for feature encoding	90
Figure 5.5	Steps involved to encode an image region	91

Figure 5.6	Illustration of height bands for pooling	93
Figure 5.7	Recursive label filtering scheme	97
Figure 5.8	Region classification performance with SIFT	102
Figure 5.9	Comparison of descriptor variants for region encoding	103
Figure 5.10	Illustration of local temporal fluctuations in the labeling decision	107
Figure 5.11	Qualitative results of our region-level scene labeling model	107
Figure 5.12	Stixmantics runtime chart	108
Figure 6.1	Texton map examples (<i>DUS</i>)	115
Figure 6.2	Texton map examples (<i>KITTI</i>)	116
Figure 6.3	Encode-and-classify tree structure	117
Figure 6.4	Final concept: training and inference	119
Figure 6.5	Influence of patch size on accuracy	121
Figure 6.6	Influence of tree depth on accuracy and histogram length	123
Figure 6.7	Runtime details	126

LIST OF TABLES

Table 3.1	Key characteristics of other datasets	35
Table 3.2	Distribution of annotations among the classes in our <i>DUS</i> dataset	42
Table 3.3	Accuracy of the static estimator (<i>DUS</i>)	46
Table 3.4	Dataset comparison by means of the static estimator	47
Table 4.1	Pixel-level confusion matrix (<i>DUS</i>)	68
Table 4.2	Pixel-level confusion matrix (<i>KITTI</i>)	69
Table 4.3	Feature channel comparison (<i>DUS</i>)	70
Table 4.4	Feature channel comparison (<i>KITTI</i>)	71
Table 4.5	Pixel classifier confusion matrix, mapped to coarse classes for Stixels	73
Table 4.6	Stixel extension results	74
Table 4.7	Extended Stixels compared to reference methods (<i>DUS</i>)	79
Table 4.8	Extended Stixels compared to reference methods (<i>KITTI</i>)	79
Table 4.9	Runtime of extended Stixels compared to reference methods	80
Table 5.1	List of bands used for height pooling	92
Table 5.2	Results of ERC _G region encoding and on ground truth segments	101

Table 5.3	Impact of using Stixels and Stixel-based regions	103
Table 5.4	Impact of descriptors and region proposals on segmentation accuracy	104
Table 5.5	Improvements based on regularization	105
Table 5.6	Quantitative results compared to reference methods (<i>DUS</i>)	109
Table 6.1	Runtime vs accuracy trade-off of our E&C trees	122
Table 6.2	Quantitative results compared to reference methods (<i>DUS</i>)	125
Table 6.3	Quantitative results compared to reference methods (<i>KITTI</i>)	125

LIST OF ALGORITHMS

4.1	Decision tree training	62
5.1	Proposal region generation from Stixels	87

ACRONYMS

ADAS	Advanced Driver Assistance Systems
ACC	Adaptive Cruise Control
ALE	Automatic Labeling Environment (Ladický et al., 2010)
AP	Average Precision
AUC	Area Under Curve
BOF	Bag-of-Features
BODF	Bag-of-Depth-Features
BOST	Bag-of-Semantic-Textons (Shotton et al., 2008)
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRF	Conditional Random Field

EM	Expectation Maximization
ERC	Extremely Randomized Clustering (Moosmann et al., 2008)
FCN	Fully Convolutional Network (Long et al., 2015)
FOE	Focus Of Expansion
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GPB	Globalized Probability of Boundary (Arbeláez et al., 2011)
GP	Global Precision
GPU	Graphics Processing Unit
HDR	High Dynamic Range
HIK	Histogram Intersection Kernel (Wu, 2010)
HMM	Hidden Markov Model
HOG	Histogram of Oriented Gradients (Dalal and Triggs, 2005)
IOU	Intersection Over Union
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago
KLT	Kanade-Lucas-Tomasi (authors of a well-known feature tracking method)
MAP	Maximum A-Posteriori
MRF	Markov Random Field
OWT	Oriented Watershed Transform
PASCAL	Pattern Analysis, Statistical Modelling and Computational Learning
RBF	Radial Basis Function
RDF	Randomized Decision Forest
ROC	Receiver Operator Characteristic
SFM	Structure From Motion
SGM	Semi Global Matching (Hirschmüller, 2008)
SIFT	Scale-Invariant Feature Transform (Lowe, 2004)
SLIC	Simple Linear Iterative Clustering (Achanta et al., 2012)

STF	Semantic Texton Forest (Shotton et al., 2008)
SVM	Support Vector Machine
UCM	Ultrametric Contour Map (Arbeláez et al., 2011)
VOC	Visual Object Classes (PASCAL challenge)

INTRODUCTION

CONTENTS

1.1	Motivation	2
1.1.1	Automotive Scene Labeling	2
1.1.2	Representing the Visual Environment . .	3
1.1.3	Combining Multiple Cues	4
1.2	Problem Statement	5
1.2.1	Why Urban Scenes?	5
1.2.2	Industrial Considerations	6
1.2.3	Approach	7
1.3	Contributions	8
1.4	Outline	9

Image analysis is quickly becoming a key technology for many practical applications. Advanced web-search, industrial quality assessment, visual aid for the blind, satellite-based monitoring of agricultural areas, and autonomous navigation are just a few examples. Also, the importance of efficient image analysis will continue to grow with the availability of cheaper image sensors and the sheer amount of data that is being generated with them. Compared to *e.g.* written text, this data is highly complex and unstructured, but yet, it contains very valuable information. The well-known idiom “A picture is worth a thousand words” summarizes this fact quite accurately and is equally true in the field of computer vision.

The topic of this dissertation is efficient scene labeling, also called semantic segmentation, with a specific focus on urban street scenes. Scene labeling is the problem of assigning a semantic class label, such as vehicle, road, or building, to every pixel in an image, as shown in Figure 1.1. While this is a research topic for many years now, there are still a lot of unsolved problems, not only in terms of accuracy but also in terms of runtime. With self-driving vehicles being on the verge of industrial realization, new, more efficient models need to be developed to support this line of research, as navigating in cluttered, man-made environments poses very high requirements in terms of perception. Scene labeling can be seen as the *analysis* part of semantic scene *understanding*, where the latter rather implies a higher level of awareness of what is happening in a depicted scene. This aspect of higher-level image understanding, although very interesting, shall not be covered in the scope of this dissertation. Instead, we focus on combining multiple different input modalities that can be extracted

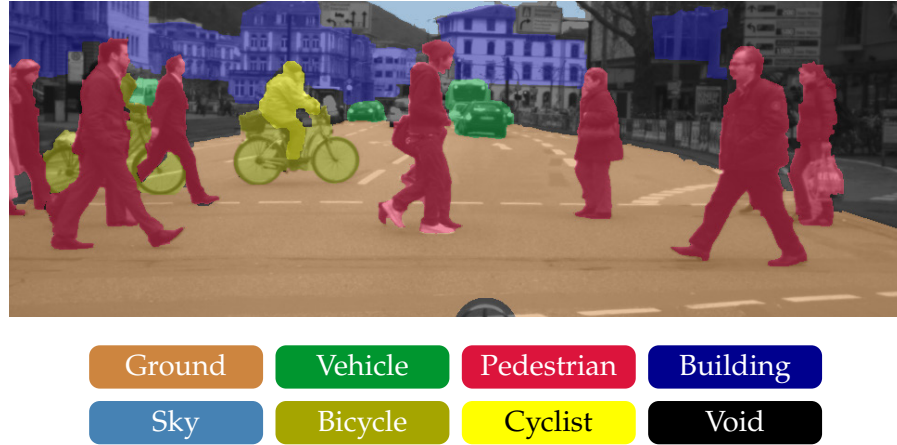


Figure 1.1: Scene labeling is the problem of assigning a class label to every pixel in the image, as shown in this example. The overlaid colors in the image indicate different labels, *i.e.* red for pedestrians, green for cars, and blue for buildings. While this example is created by manual annotation, the goal of scene labeling is to infer this information automatically, without human assistance.

from video data, namely appearance, depth, and motion; analyze which kind of information is most valuable for which subproblem and propose a model to combine them in an efficient way.

1.1 MOTIVATION

1.1.1 Automotive Scene Labeling

Over the last years, there has been a steady increase of Advanced Driver Assistance Systems (ADAS) in vehicles of many manufacturers, from traffic sign recognition and lane departure warning to Adaptive Cruise Control (ACC), automatic lane keeping, and fully autonomous braking for vulnerable road users such as pedestrians. Many of these new features are only possible thanks to novel sophisticated algorithms to process the data and image analysis plays an important part in that, due to the versatility and comparably cheap price point of cameras. However, most of these features are specifically tailored for highway scenarios and cannot easily be transferred to inner-city traffic situations, because of inherently different demands on the system capabilities in the two scenarios. On a highway, the high speed of the vehicle imposes very strict requirements in terms of detecting obstacles far ahead, but the environment is comparably structured, *i.e.* only few different object classes are relevant to control the vehicle. This allows a step-wise integration of features, *e.g.* adding detection of a new object class to handle a specific scenario better, where every feature is implemented as a separate, independent module. In inner-city traffic, it is the other way around. The lower velocity re-

laxes the hard constraints with respect to detection range, but the environment is very complex, *i.e.* a large number of object classes is relevant for navigation, so that step-wise integration is not an option. Instead, there needs to be a scalable approach that infers a multitude of object categories at the same time: traffic lights and signs to follow the rules of traffic, poles, tree trunks, road markings and buildings for localization, all kinds of dynamic and static obstacles that need to be avoided, grass and sidewalk to delimit the drivable space in case no lane markings are present. In this context, scene labeling is a very interesting option, as image regions and their semantic class label are inferred simultaneously within the same method, rather than applying individual object detectors for every class. This is particularly important in terms of computational resources and power consumption, both of which are traditionally limited in a vehicle: scene labeling allows to share intermediate results between different object classes. Furthermore, many objects in inner-city traffic are partially occluded, so that prior knowledge about the shape of an object is less informative. Again, scene labeling methods are rather generic in this sense. We therefore argue that the new challenges we are facing in terms of visually understanding complex urban traffic environments can only be tackled successfully with scene labeling as a central component in automotive vision systems.

1.1.2 Representing the Visual Environment

Efficiently extracting relevant information from images is one side. The other side is to represent this data in a way that makes it easy and efficient for subsequent processing stages to work with the extracted information. Naturally, those two sides are coupled to a certain degree, but it is not well defined what the ideal representation would look like. In the literature, recent years have seen a steady progression from geometrically rigid (bounding box) classification methods, *e.g.* (Dalal and Triggs, 2005; Keller et al., 2011), through deformable part models, *e.g.* (Felzenszwalb et al., 2010), towards the classification of free-form regions in the image, *e.g.* (Carreira et al., 2012; Fulkerson et al., 2009; Ladický et al., 2010; Shotton et al., 2009).

Object-centric approaches are designed to detect instances of one particular object class in an image. Object detectors traditionally search the image in a sliding window fashion (Dollár et al., 2012; Felzenszwalb et al., 2010), where the search window has a specific aspect ratio that is fixed and matches a particular class of interest, *i.e.* 2:1 for a pedestrian and 1:2 for a vehicle side perspective. Feature descriptors typically used in this context are based on HOG, SIFT, and more recently also deep CNNs. Such detectors have shown remarkable recognition performance due to strong scene and geometric model constraints (holistic or deformable bounding-boxes), easy cue inte-

gration and strong temporal tracking-by-detection models. More recently, novel approaches such as R-CNN (Girshick et al., 2014) and YOLO (Redmon et al., 2016), certainly relaxed the strict aspect ratios and methods to generate hypotheses, but the target of fitting a bounding box around objects is still the same. As a result, the scene content is represented very concisely as a set of individual detected objects. However, the generalization to partial occlusion cases, object groups or classes that do not have a well-defined geometric structure, such as road surface or building, is difficult.

Region-centric models, i.e. (semantic) segmentation approaches, such as (Carreira et al., 2012; Fulkerson et al., 2009; Shotton et al., 2009; Wojek and Schiele, 2008) among many others, operate in a bottom-up fashion and usually do not recover an object-level scene representation. They are rather generic in terms of the geometry and the number of object classes involved. In this context, both dense features and orderless models (BOF) have proven very effective (Zhang et al., 2006), consisting of: local feature extraction within a free-form region, coding and spatial pooling of all features within the given region, and subsequent discriminative classification of the pooled feature vector. However, grouping is based on pixel-level intensity, depth or motion discontinuities with only few geometry or scene constraints. This lack of constraints can easily lead to errors in the recovered scene model. Furthermore, representing the environment in terms of labeled pixels, as scene labeling per se does, is overly redundant for most applications. We therefore argue for a model that takes advantage of; or is in-between both approaches: A model neither too specific nor too generic that represents the surroundings in a compact and efficient way.

1.1.3 Combining Multiple Cues

Obstacle detection can be performed based on different sources of image information. Motion, so-called optical flow, can be used to focus on potential hazards independently of their object category, which is usually referred to as detection-by-tracking. Vice versa, motion can also help to check the plausibility of a detection over time, which is then referred to as tracking-by-detection. However, in a dynamic environment, where not only the observing platform but also other obstacles are moving, motion information alone is not sufficient, as it is unclear what the cause of the apparent motion in the image is. Stereo depth information is similar to motion in terms of the ability to detect generic obstacles without any specification of object classes. Also, as two independent cameras record an image at the same time, dynamic objects can be detected explicitly as moving and compared to the two-dimensional matching problem for optical flow, stereo estimation is a one-dimensional matching problem, which typically al-

lows more robust and precise estimation. Another important benefit of (calibrated) stereo information is that it provides metric distance and thus also adds a measure of object scale. However, both motion and stereo are limited in terms of identifying different types of objects. This is where image appearance and classification models can help. Learning-based approaches are very good at differentiating various types of objects based on their visual appearance in the image. On the other hand, pure appearance-based models are often computationally expensive, as they need to compensate the missing knowledge about the depicted scene structure and scale that is available from motion and stereo by more exhaustive search.

In summary, every cue has its particular benefits and drawbacks, so we argue that a holistic representation of the environment should combine these orthogonal cues to maximize individual benefits while minimizing individual drawbacks. However, rather than using all available information at all times, it is important to understand which cue is most valuable in which situation, so as to allow a targeted combination that still allows efficient execution.

1.2 PROBLEM STATEMENT

Based on the previous discussion, the problem considered in this dissertation is to develop a scene labeling concept for outdoor urban street scenes that is well-suited for application in an automotive setting. This concept needs to fulfill several conditions: (1) it needs to be scalable to a large number of object classes and at the same time it needs to be computationally efficient enough to run in real-time, with typical camera refresh rates, (2) it needs to represent the environment in a compact way that is neither too specific nor too generic, so that subsequent processing stages have efficient access to the generated information, (3) it should combine appearance, depth and motion cues so that they complement each other in an efficient way, (4) it should be able to handle partial occlusion cases that often occur in inner-city traffic, and finally (5) the concept needs to be competitive to reference methods in the literature.

1.2.1 *Why Urban Scenes?*

In fact, urban traffic is not only an interesting testbed for research because self-driving cars are about to enter this domain. The complexity of these scenes is also challenging for existing approaches. For instance: One question this dissertation aims to answer is to what extent depth information can contribute to the solution of the scene labeling problem. In particular, it is interesting to find out if algorithms that make use of depth data are able to increase the overall accuracy of the result, if they yield similar results to state-of-the-art monocular

approaches but at reduced computational costs or if they can even deliver both at the same time: better quality *and* lower computation time. An urban scenario is well suited as a testbed to answer this question. Compared to the relatively structured street layout of highways and rural roads, urban traffic is highly complex and cluttered, with many kinds of obstacles and traffic participants at strongly varying scales. This makes it a very challenging task to find all objects of interest in the scene from a single image. The open question is, in how far the additional structure that can be obtained from depth cues can help to overcome or relax this difficulty.

Another problem we are seeking to address in this dissertation is detection under strong occlusion. Again, urban streets are a good scenario for this, as they are often packed with cars driving or parking on the side of the road. In fact, most of the objects in the scene are not fully visible but mutually occlude each other, so that classical bounding box detectors with a rigid appearance model have difficulties to detect them. Finally, the concepts presented in this dissertation are all developed with fast execution time in mind, as algorithms built to support autonomous navigation need to adapt quickly to new situations. Inner-city traffic is one of the most dynamic environments one can possibly imagine. Consequently, only very limited assumptions on the dynamics of objects can be imposed, which asks for methods that reason instantly and avoid overly strong model assumptions. In light of the aforementioned aspects, we find that the setting of inner-city urban scenes allows best to address the problems we are facing throughout this dissertation. Later in this work, we will briefly introduce the most relevant datasets with focus on outdoor urban scene labeling that have been presented in the literature so far and discuss their individual properties.

1.2.2 *Industrial Considerations*

The work presented in this dissertation was prepared under joint supervision of the *Image Understanding Group* at Daimler AG and the *Visual Inference Group* at TU Darmstadt. With Daimler being the main financial sponsor of this work, all design choices have been developed with efficient execution time in mind to allow real-time in-vehicle application. This also includes a tight integration of the method into a large framework of existing algorithms for depth estimation, segmentation and tracking. While this fact seems to be a minor side note at first glance, it comes along with an important implication. Many approaches in the literature deal with the topic of jointly estimating multiple correlated parameters, *e.g.* as depth and semantic labels (Kundu et al., 2014; Häne et al., 2013; Ladický et al., 2010), or feeding back higher-level results to lower-level algorithms to improve an initial estimate. While these strategies are perfectly reasonable from a

conceptual point-of-view, they often introduce practical problems. Estimating parameters jointly for instance hinders modularization and feedback loops in a system architecture can easily introduce mutual dependencies between software components. In terms of functional safety and module testing, a streamlined processing chain of smaller logical blocks is therefore usually preferred. The existing framework at Daimler contains excellent methods for solving individual problems such as depth estimation and optical flow that are highly optimized for automotive applications. We therefore aim to develop a model for scene labeling that benefits from these methods rather than potentially replacing them. This includes both depth maps and sparse point tracks, which we assume as given throughout this dissertation and build our model around it.

1.2.3 Approach

This dissertation proposes a scene labeling model that is well-suited for application in an intelligent vehicle. The thesis put forward in this dissertation is that such a scene labeling model needs to be modular and needs to make re-use of computational resources as much as possible to be practically feasible. The modularity aspect is rather related to the general system architecture of an intelligent vehicle and how scene labeling can be incorporated in such a system without major use of additional resources. This is different compared to existing work, which often relies on resource-intensive models and merely focuses on the quality of the result. In this dissertation, the focus is not only to provide good scene labeling performance, but also how scene labeling can be efficiently embedded into a complex architecture in an intelligent vehicle. Therefore, we propose a modular architecture for scene labeling that consists of several components such as stereo depth map computation, pixel-level classification, region generation and classification, feature tracking, temporal label integration, and CRF-based regularization. Each individual component is fast to compute, leverages information from preceding components if it supports the task, and provides additional information as much as it is capable of, not less and not more. In doing so, we follow the principle of least commitment. According to this principle, it is better to wait with decisions until enough knowledge is gained. In other words, we divide the scene labeling problem into a sequence of simpler subproblems that can be solved efficiently. The idea is to find feasible solutions for each subproblem and at the same time to combine the results in a good way, so that potential errors of individual building blocks can be resolved in the overall systems. As level of representation we choose the well-established *Stixel World* model of Pfeiffer (2011). Building the modules proposed in this dissertation around this model allows to reduce inference time and also increases the compactness and expres-

siveness of the result compared to a dense label image. More details about the contributions of this work are given in the next section.

1.3 CONTRIBUTIONS

In light of the given problem statement, the industrial context this work is embedded in and the shortcomings of existing approaches in terms of either runtime, level of representation or combination of multiple input modalities that we will discover in the next chapter, the main contributions of this dissertation are: (1) integration of appearance cues into the Stixel model, (2) Stixel-based region generation and classification, (3) tracking for temporally consistent scene labeling, (4) a joint model for efficient encoding and classification, and (5) a novel stereo vision dataset with semantic labels. In the following we will discuss the contributions in more detail.

INTEGRATION OF APPEARANCE INTO THE STIXEL MODEL The Stixel framework as introduced in Section 2.2 has found wide adoption in the automotive industry and academia alike. Given that the existing model solely reasons based on depth information, we provide a framework to integrate appearance into the estimation process. The integration of appearance and depth should ultimately lead to a model with improved robustness under challenging conditions. To this end, we propose a low-level classification scheme that integrates multiple cues and extend the mathematical formulation of the Stixel model to take this new information into account.

STIXEL-BASED REGION GENERATION AND CLASSIFICATION We follow the recent trend of scene labeling approaches that classify large bottom-up proposal regions, as they have high spatial support and hence carry a lot of information to learn strong classification models. In contrast to previous work, we utilize our augmented Stixel model to rapidly generate and classify proposal regions. For classification, we follow a Bag-of-Features paradigm, discuss how the feature encoding step in this paradigm can be improved in terms of runtime and accuracy and propose a novel pooling strategy that introduces a 3D geometrical ordering into the region descriptor.

TEMPORALLY CONSISTENT SCENE LABELING To leverage temporal information available from video, we propose a novel way to regularize a dense scene labeling result temporally by means of sparse long-term point trajectories. Our integration scheme is formalized as a Bayesian recursive labeling problem for each individual point track. The reasoning on the trajectory level allows to set aside the registration problem of superpixels between different frames, which is one of the central issues in common approaches.

A JOINT MODEL FOR ENCODING AND CLASSIFICATION Based on the two competing approaches of pixel-level and region-level scene labeling we will present in this dissertation, we propose a model to combine both approaches. The proposal is supported by an experimental evaluation that shows which factors are important for good performance of the individual methods.

A NOVEL STEREO VISION DATASET Given that most existing scene labeling datasets do not provide depth information or are too small in terms of resolution and size, we introduce a challenging novel dataset for outdoor urban scene labeling that comes with video, stereo imagery, pre-computed depth maps and camera calibration, and semantic class annotations into seven classes. Our dataset consists of 5000 stereo image pairs, where 500 images have been labeled by a human annotator.

1.4 OUTLINE

The remainder of this dissertation is structured as follows. Chapter 2 covers the technical background and competing approaches in the literature, which are related to the methods presented in this work. As scene labeling involves machine learning, it heavily depends on the availability of annotated data. Chapter 3 therefore discusses existing datasets and introduces a novel dataset for outdoor urban scene labeling that provides all information required to investigate the questions this dissertation tries to answer. The main technical contributions are then split into three parts. Part I (Chapter 4) approaches the scene labeling problem as classification of small local images patches, based on dense features channels extracted from multiple cues and further introduces an extension to Stixels, to take semantics into account during inference. Parts of this work have previously been published in (Scharwächter and Franke, 2015) and (Cordts et al., 2017b). Based on the findings in that chapter, Part II (Chapter 5) then investigates an alternative method that tackles scene labeling as classification of larger bottom-up region proposals. Furthermore, we propose different ways of improving accuracy and runtime of such a method in case depth information is available. The work presented in this chapter has previously been published in (Scharwächter et al., 2013) and (Scharwächter et al., 2014). Part III (Chapter 6) finally introduces a framework to combine the benefits of both investigated approaches into a single model, which has been published in (Cordts et al., 2017a). Chapter 7 provides an in-depth discussion of the presented work and concludes the dissertation.

CONTENTS

2.1	Reasoning from Motion and Stereo	11
2.2	The Stixel World	14
2.2.1	Graphical Model	15
2.2.2	Inference	19
2.3	The Bag-of-Features Model	19
2.3.1	Feature Encoding Methods	20
2.3.2	Pyramidal Encoding	21
2.4	Scene Labeling	22
2.4.1	Pixel- and Region-level Scene Labeling	23
2.4.2	Temporally Consistent Scene Labeling	24
2.4.3	Scene Labeling Reference Methods	25

This chapter covers the technical foundations of this dissertation and summarizes work from the literature that is strongly related to methods proposed in this work. Rather than directly entering the vast body of literature on semantic segmentation and scene labeling, we approach it from an automotive perspective, starting with methods and concepts applied around the time this dissertation was started. We then introduce techniques for image segmentation, including an in-depth introduction to the so-called Stixel world, as well as feature encoding methods that are frequently used in the context of scene labeling. Finally, we will look into specific aspects of scene labeling that are relevant for this work and point out shortcomings we encountered in existing work.

2.1 REASONING FROM MOTION AND STEREO

Around the time this dissertation was started, there were mainly two individual streams of vision research conducted in the automotive industry: learning-based bounding box object detectors for specific classes such as pedestrians, vehicles or traffic signs, and generic bottom-up models of the environment based on motion tracking and stereo reconstruction. While we already discussed the abstract differences of those two types of environment representations in Section 1.1.2, we now present some bottom-up methods that have been developed prior to this dissertation: 6D-Vision, Stixels, and Stixmentation. In some respect, those methods denote the starting point of



Figure 2.1: Example images of the 6D-Vision approach that estimates 3D velocity of individual points in the image. Arrows show the predicted position of a point 500 ms in the future. Points without an arrow are stationary. Color indicates distance from near (red) to far (green). Courtesy of Rabe.

the work presented in this dissertation and are relevant in order to understand the rationale for some of the design decisions made later in this work.

Following the classical detection-by-tracking principle, the *6D-Vision* approach by [Franke et al. \(2005\)](#) detects motion of dynamic obstacles in the scene irrespective of the object category. This is achieved by tracking and filtering the position and velocity of individual points over time by means of Kalman filters (one filter per point). Apparent motion induced by movement of the observing vehicle is compensated to recover absolute motion in 3D Euclidean space. The concept relies on sparse point trajectories that are initialized at strong image corners and tracked with a Kanade-Lucas-Tomasi ([KLT](#)) feature tracker. Depth information for each tracked point is provided by disparity maps, which are computed in advance by means of Semi Global Matching ([SGM](#)) ([Hirschmüller, 2008](#)). The 6D-Vision approach is very general, as it allows to detect arbitrary motion, and has been shown to work very well in practice. However, the method can fail in cases where image texture is insufficient or oriented along the Focus Of Expansion ([FOE](#)), such as on guard rails on the highway, or due to reflections in windows or on vehicles, because context is not taken into account. These problems can only be solved by adding semantic information into the estimation process. Examples of the 6D-Vision approach are shown in [Figure 2.1](#).

The *Stixel World*, introduced by [Badino et al. \(2009\)](#) and later extended by [Pfeiffer \(2011\)](#) has been developed to compress 3D point clouds into a set of so-called Stixels that represent the 3D environment around a vehicle as a set of sticks: upright standing obstacles that are oriented perpendicular to a supporting ground plane. Based on the assumption that man-made environments are typically rather structured, this model allows to reduce the number of parameters necessary to describe the scene content and at the same time regularizes noisy depth measurements. The Stixel model plays a central role in this dissertation and will be introduced in more detail later in Sec-

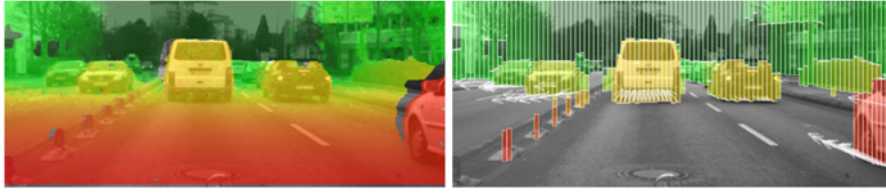


Figure 2.2: Dense stereo depth map computed with Semi Global Matching (SGM) (left) and resulting Stixel representation (right). Colors indicate distance from near (red) to far (green) and the arrow at the bottom of each Stixel indicates its motion direction. Courtesy of Pfeiffer.

tion 2.2, together with a discussion about drawbacks of the existing model that can be alleviated by making use of semantics. Stixels can be seen as an intermediate step between pixel level and object level and can be used as primitive elements for higher-level reasoning to improve efficiency, similar to what superpixels are used for. Figure 2.2 shows Stixels for a typical traffic scene, where color encodes distance.

Combining the two previous concepts then allows to estimate the velocity of each Stixel, as shown by Pfeiffer and Franke (2011b). Using these velocity-augmented Stixels as input, the *Stixmentation* approach of Erbs et al. (2012) segments the scene into discrete motion classes (forward-moving, oncoming, left-moving, right-moving) and static infrastructure by formulating a Conditional Random Field (CRF) model on top of Stixels that takes into account 3D position and motion features as well as spatio-temporal prior knowledge. This chain of algorithms that build up a rich environmental model, starting at pixels, estimating 3D motion and Stixels, clustering Stixels into motion groups, and eventually inferring objects has also been summarized in Pfeiffer et al. (2012). More examples of Stixels and the inferred Stixmentation results are shown in Figure 2.3.

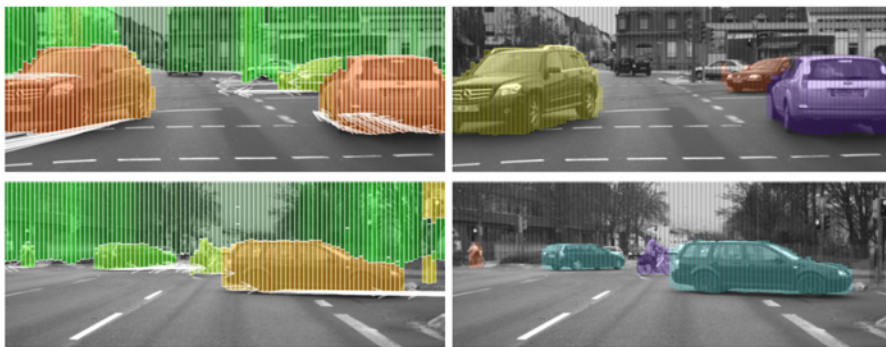


Figure 2.3: Stixels (left) and motion clusters inferred by the Stixmentation approach (right). Black Stixels are static infrastructure and yellow, purple, orange and turquoise indicate four different discrete motion directions. Courtesy of Pfeiffer and Erbs.

In this dissertation, we follow this multi-year effort of incrementally building up a Stixel-based environment representation by extending it with appearance-based semantic information. This will not only help to increase the expressiveness of the model but also to resolve problems that cannot be resolved by using depth and motion information alone.

2.2 THE STIXEL WORLD

This section presents the original concept of Stixels as introduced by Pfeiffer (2011) that only relies on dense depth maps as input, but with a novel mathematical notation introduced by Marius Cordts in (Cordts et al., 2017b). We will follow this notation later in Section 4.3, when we extend the Stixel concept with semantic label information. The following text is re-written in my own words based on (Cordts et al., 2017b), where Marius Cordts, Timo Rehfeld (myself) and Lukas Schneider contributed equally.

The Stixel World \mathcal{S} is similar to superpixels in the sense that it segments an image into a set of small primitive elements. However, in comparison to most other segmentation approaches, which treat segmentation of a $w \times h$ image as a two-dimensional problem, the Stixel algorithm breaks this down to $\frac{w}{w_s}$ individual one-dimensional problems, one for each column of width w_s . The parameter $w_s \geq 1$ controls the fixed horizontal extent of each Stixel and can be seen as discretization factor that reduces computational burden. In contrast, the height of each Stixel, *i.e.* its vertical extent, is the essential free parameter that is inferred explicitly in the model. To have more control over inference complexity, we also introduce an optional vertical sub-sampling factor $h_s \geq 1$.

The central argument for the proposed Stixel model is based on the observation that street scenes have a strong repetitive structure along the vertical direction, *i.e.* along one column in an image (irrespective of the horizontal position of the column). Much stronger than along the horizontal direction. The Stixel model leverages this fact during inference by focusing on the more relevant vertical structure and ignoring horizontal structure across neighboring columns, which allows to infer all columns independently and in parallel. We denote Stixels as *medium-level* representation for three reasons: (1) in contrast to pixels, Stixels provide a more expressive notion of depth and physical object extent; (2) in contrast to bottom-up superpixels, Stixels are inferred based on prior knowledge about the typical street scene layout; (3) in contrast to an object-level representation, Stixels do not provide a notion of physical instance, because one object is typically covered by several Stixels horizontally. Therefore, Stixels can be seen as a compressed representation that allows subsequent higher-level algorithms to parse the scene efficiently.

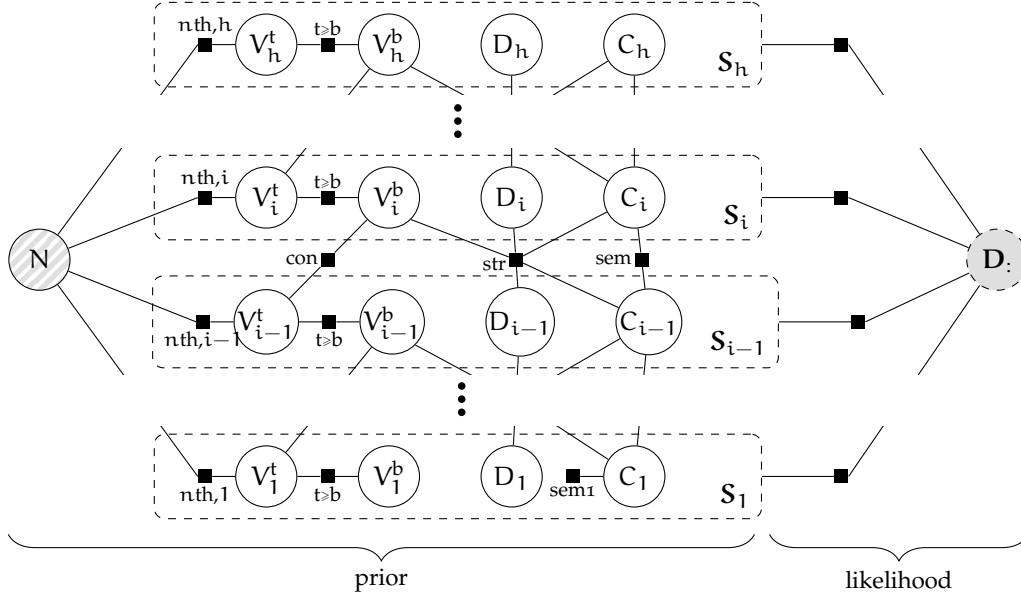


Figure 2.4: The Stixel world as a factor graph, showing the factorization of Equation (2.1). Each Stixel S_i (dashed boxes) is described by four random variables V_i^t, V_i^b, C_i, D_i (circles). The hatched node on the left denotes the random variable N describing the number of Stixels n constituting the final segmentation. Black squares denote factors of the posterior (see descriptions in the text). The circle $D_?$ on the right denotes the measurements, *i.e.* a column of the disparity map. Figure adapted from (Cordts et al., 2017b). Courtesy of Marius Cordts.

Each Stixel has a class label and a planar 3D depth model attached. Following the definition of Pfeiffer (2011), the label set consists of three *structural classes*: “support” (\mathcal{S}), “vertical” (\mathcal{V}), and “sky” (\mathcal{Y}), which can be separated solely based on the geometric assumptions of the underlying scene model: support Stixels have a constant height offset relative to a predetermined ground model, vertical Stixels have a constant depth and the depth of sky Stixels is infinite or invalid. The following section provides the mathematical details of the Stixel model for a single image column of width w_s .

2.2.1 Graphical Model¹

We start by introducing the posterior distribution $P(S_? | M_?)$ of a Stixel column $S_?$, conditioned on all measurements $M_?$ within the column. Following Pfeiffer (2011), $M_?$ is identical to one column of the disparity map $D_?$ obtained via stereo matching. The above posterior is defined with a graphical model, which is depicted as a factor graph

¹ This section is original work of Marius Cordts and added in my own words here for completeness.

in Figure 2.4. This graph contains a factorization of the posterior into a likelihood term $\tilde{P}(\mathbf{M}_: | \mathbf{S}_:)$ and a prior term $\tilde{P}(\mathbf{S}_:)$, yielding

$$P(\mathbf{S}_: | \mathbf{M}_:) = \frac{1}{Z} \tilde{P}(\mathbf{M}_: | \mathbf{S}_:) \tilde{P}(\mathbf{S}_:) , \quad (2.1)$$

where Z is the partition function for normalization. Note that both, likelihood and prior are unnormalized probability mass functions. In the log-domain, this yields

$$P(\mathbf{S}_: = \mathbf{s}_: | \mathbf{M}_: = \mathbf{m}_:) = e^{-E(\mathbf{s}_:, \mathbf{m}_:)} , \quad (2.2)$$

where $E(\cdot)$ is the energy function, which is defined as

$$E(\mathbf{s}_:, \mathbf{m}_:) = \Phi(\mathbf{s}_:, \mathbf{m}_:) + \Psi(\mathbf{s}_:) - \log(Z) . \quad (2.3)$$

The function $\Phi(\cdot)$ corresponds to the likelihood and $\Psi(\cdot)$ to the prior.

In order to correctly define all energies, a fixed number of random variables is required to describe the segmentation $\mathbf{S}_:$. This is achieved by splitting the column $\mathbf{S}_:$ into the maximum number of possible Stixels \mathbf{S}_i , *i.e.* $i \in \{1 \dots h\}$, and adding an extra random variable $N \in \{1 \dots h\}$ that denotes the actual number of inferred Stixels within a column. For a certain value n , all factors connected to a Stixel \mathbf{S}_i with $i > n$ are then set to zero energy, so that these factors do not influence the resulting segmentation at all. For ease of notation, we assume that $i \leq n$, so that the dependency of the factors on N can be dropped.

The column segmentation $\mathbf{S}_:$ consists of h Stixels \mathbf{S}_i , with \mathbf{S}_1 being the lowest and \mathbf{S}_h being the highest Stixel. A Stixel \mathbf{S}_i in turn splits into four random variables V_i^b , V_i^t , C_i , and D_i . The first two denote the vertical extent (bottom and top). The variable C_i represents the class label, and D_i parameterizes the respective disparity model: for vertical Stixels the constant disparity, for support Stixels the constant disparity *offset* to the ground plane.

2.2.1.1 Prior

The prior $\Psi(\mathbf{s}_:)$ from Equation (2.3) captures knowledge about the scene structure independent of any measurements. It factorizes as

$$\Psi(\mathbf{s}_:) = \Psi_{mc}(n) + \sum_{i=1}^h \sum_{id} \Psi_{id}(\mathbf{s}_i, \mathbf{s}_{i-1}, n) , \quad (2.4)$$

where *id* stands for the factor name in Figure 2.4. Note that not all factors actually depend on all variables \mathbf{s}_i , \mathbf{s}_{i-1} , or n . In the following, all individual factors are defined in more detail: model complexity, segmentation consistency, structural priors, and semantic priors.

MODEL COMPLEXITY The model complexity prior Ψ_{mc} is the essential regularization term and controls the compromise between compactness and accuracy. The factor is defined as

$$\Psi_{\text{mc}}(n) = \beta_{\text{mc}} n . \quad (2.5)$$

The higher the parameter β_{mc} , the smaller the number of Stixels, which in turn results in stronger regularization.

SEGMENTATION CONSISTENCY We add hard constraints to ensure that segments are non-overlapping, connected, and extend over the whole image, *i.e.* the first Stixel must begin in image row 1 (bottom row) and the last Stixel must end in row h (top row):

$$\Psi_{1\text{st}}(v_1^b) = \begin{cases} 0 & \text{if } v_1^b = 1 \\ \infty & \text{otherwise} \end{cases} , \quad (2.6)$$

$$\Psi_{\text{nth},i}(n, v_i^t) = \begin{cases} \infty & \text{if } n = i \text{ and } v_i^t \neq h \\ 0 & \text{otherwise} \end{cases} . \quad (2.7)$$

Furthermore, consecutive Stixels must be connected and the top point must always be larger than the bottom point of each Stixel, *i.e.*

$$\Psi_{\text{tb}}(v_i^b, v_i^t) = \begin{cases} 0 & \text{if } v_i^b \leq v_i^t \\ \infty & \text{otherwise} \end{cases} , \quad (2.8)$$

$$\Psi_{\text{con}}(v_i^b, v_{i-1}^t) = \begin{cases} 0 & \text{if } v_i^b = v_{i-1}^t + 1 \\ \infty & \text{otherwise} \end{cases} . \quad (2.9)$$

STRUCTURAL PRIORS We encode the typical 3D layout of road scenes with Ψ_{str} , which is in turn comprised of two factors, both responsible for an individual effect, *i.e.*

$$\Psi_{\text{str}}(c_i, c_{i-1}, d_i, d_{i-1}, v_i^b) = \Psi_{\text{grav}} + \Psi_{\text{do}} . \quad (2.10)$$

The gravity component Ψ_{grav} is only non-zero for $c_i \in \mathcal{V}$ and $c_{i-1} \in \mathcal{S}$ and models that a vertical Stixel is usually not floating but instead standing on the preceding support surface Stixel. Consequently, the disparity of the vertical Stixel d_i and the disparity of the preceding support Stixel must coincide in row v_i^b . The latter disparity is denoted by the function $d_s(v_i^b, d_{i-1})$ and their difference as $\Delta_d = d_i - d_s(v_i^b, d_{i-1})$. Then, Ψ_{grav} is defined as

$$\Psi_{\text{grav}} = \begin{cases} \alpha_{\text{grav}}^- + \beta_{\text{grav}}^- \Delta_d & \text{if } \Delta_d < 0 \\ \alpha_{\text{grav}}^+ + \beta_{\text{grav}}^+ \Delta_d & \text{if } \Delta_d > 0 \\ 0 & \text{otherwise} \end{cases} , \quad (2.11)$$

where α_{grav} and β_{grav} are free parameters of the model. The second term Ψ_{do} evaluates the relative depth ordering of vertical segments.

Usually, an object that is visually located on top of another object in the 2D image is farther away in the 3D scene, *i.e.* the top disparity is smaller than the bottom one, which is encoded as

$$\Psi_{\text{do}} = \begin{cases} \alpha_{\text{do}} + \beta_{\text{do}} (d_i - d_{i-1}) & \text{if } c_i \in \mathcal{V}, c_{i-1} \in \mathcal{V}, d_i > d_{i-1} \\ 0 & \text{otherwise} \end{cases} . \quad (2.12)$$

SEMANTIC PRIORS The last factor Ψ_{sem} incorporates a-priori knowledge about the semantic structure of road scenes:

$$\Psi_{\text{sem}}(c_i, c_{i-1}) = \gamma_{c_i} + \gamma_{c_i, c_{i-1}} . \quad (2.13)$$

The first term γ_{c_i} encodes the energy of the respective a-priori probability of class c_i , where a higher value means that a Stixel with that class label is less likely. The second term $\gamma_{c_i, c_{i-1}}$ is defined via a two-dimensional matrix $\gamma_{c_i, c_{i-1}}$, rating all possible combinations of class transitions. Entries in this matrix model expectations about relative class locations, *e.g.* a vertical Stixel above a support Stixel is rated more likely than vice versa. Note that only relations of first order are captured to allow for efficient inference. Finally, $\Psi_{\text{sem1}}(c_1)$ encodes the class prior of the very first Stixel, which is more likely “support” than “vertical” or “sky”.

2.2.1.2 Data likelihood

The likelihood term from Equation (2.3) rates the impact of measurements, in our case the disparity map \mathbf{D} , and factorizes as

$$\Phi(\mathbf{s}, \mathbf{m}) = \sum_{i=1}^h \sum_{v=v_i^b}^{v_i^t} \Phi_{\mathbf{D}}(\mathbf{s}_i, d_v, v) . \quad (2.14)$$

Note that this sum iterates over the maximum number of Stixels h , but as discussed above, all factors for $i > n$ are set to zero. Furthermore, we assume that the data likelihoods for a given Stixel segmentation \mathbf{s} are independent across pixels, so that their contribution decomposes over the rows v .

According to the Stixel world model, the depth likelihood terms consist of supporting and vertical planar surfaces. Given that the width w of Stixels is rather small, the influence of horizontal slanted surfaces is neglected and instead, with some discretization, represented via neighboring Stixels at varying depths. In doing so, the 3D orientation of a Stixel is sufficiently described by the structural class, *i.e.* support or vertical. Consequently, the 3D depth model of a Stixel is parametrized by a single variable D_v paired with its 2D position in the image and its class label. This variable is the Stixel’s constant disparity for a vertical segment and a constant disparity offset relative to the ground plane for a support segment.

We use dense disparity maps as depth measurements, where each pixel has either a disparity value assigned or is marked as invalid, *i.e.* $d_v \in \{0 \dots d_{\max}, d_{\text{inv}}\}$. The subscript v denotes the row index within the considered column. The depth likelihood term $\Phi_D(s_i, d_v, v)$ is derived from a generative measurement model $P_v(D_v = d_v \mid S_i = s_i)$ according to

$$\Phi_D(s_i, d_v, v) = -\delta_D(c_i) \log(P_v(D_v = d_v \mid S_i = s_i)) . \quad (2.15)$$

This term includes class-specific weights $\delta_D(c_i)$ that allow to learn the relevance of the depth information for a certain class. Now let p_{val} be the prior probability of a valid disparity measurement. We define

$$P_v(D_v \mid S_i) = \begin{cases} p_{\text{val}} P_{v,\text{val}}(D_v \mid S_i) & \text{if } d_v \neq d_{\text{inv}} \\ (1 - p_{\text{val}}) & \text{otherwise} \end{cases} , \quad (2.16)$$

where $P_{v,\text{val}}(D_v \mid S_i)$ denotes the measurement model of valid measurements only and is defined as a mixture of a uniform and a Gaussian distribution:

$$P_{v,\text{val}}(D_v \mid S_i) = \frac{p_{\text{out}}}{Z_U} + \frac{1 - p_{\text{out}}}{Z_G(s_i)} e^{-\frac{1}{2} \left(\frac{d_v - \mu(s_i, v)}{\sigma(s_i)} \right)^2} . \quad (2.17)$$

While the Gaussian describes the noise model of disparity measurements with standard deviation σ , the uniform distribution increases robustness against outliers and is weighted by p_{out} . The Gaussian is centered at the disparity value $\mu(s_i, v)$ of the Stixel s_i , which is constant for vertical Stixels, *i.e.* $\mu(s_i, v) = d_i$, and depends on row v for support Stixels to follow the pre-determined ground model. The parameters p_{val} , p_{out} , and σ are either chosen a-priori, or can be obtained via confidence estimation in the stereo matching algorithm, as shown *e.g.* by Pfeiffer et al. (2013).

2.2.2 Inference

Inference is performed by finding the Maximum A-Posteriori (MAP) solution that maximizes Equation (2.1), or equivalently by minimizing the energy function in Equation (2.3). One motivation for this is that compared to a maximum marginal estimate, the obtained segmentation is consistent in terms of the constraints described in Section 2.2.1.1. More details on the inference algorithms and approximations made to improve inference time can be found in (Cordts et al., 2017b).

2.3 THE BAG-OF-FEATURES MODEL

Bag-of-Features (BOF) is a common principle to encode a set of feature descriptors into a histogram of N representative descriptors. Originally introduced in the context of natural language processing, where

the concept is known as *bag-of-words* (Lewis, 1998), the objective is to find a representation that allows to compare text bodies of different and arbitrary lengths. This is done by disregarding the order of words, *i.e. throwing them all into a bag*, and just counting the number of occurrences in the text. The resulting histogram can then be used to classify documents, *e.g.* into different topics. The main difference between bag-of-words and bag-of-features is an intermediate quantization step. For text, every word is a unique entity already, so building a histogram over a pre-defined dictionary is straight-forward. For images on the other hand, color and texture patterns are typically encoded by high-dimensional feature descriptors in continuous space, so they need to be quantized to a best matching unique representative descriptor before they can be added to a histogram. We briefly discuss popular methods of vector quantization in Section 2.3.1.

We use the terms
feature encoding and
vector quantization
interchangeably in
this dissertation.

In Chapter 5, the scene labeling problem is treated as classification of bottom-up region proposals, where the BOF model is used to encode the appearance of a region. Encoding free-form image regions with bounding-box classification models such as HOG can be difficult, as they additionally capture background clutter that is not part of the actual region. This introduces higher variance in the descriptor and thus requires more training samples to yield a good model. The BOF allows to encode regions of arbitrary size and shape without this problem. Furthermore, classical models such as HOG take into account the spatial layout of texture patterns within a bounding box, which leads to strong classification results under full visibility but poor performance under occlusion. While this problem has been solved partially with the revival of deep CNNs, the BOF model does not suffer from this problem in general, as it is inherently orderless, *i.e.* the position of a texture pattern within the region is not taken into account.

2.3.1 Feature Encoding Methods

K-MEANS The most popular approach to quantize image descriptors to *visual words* is k-Means, as originally shown by Csurka et al. (2004). The k-Means algorithm finds k cluster centers that approximate the underlying data distribution in feature space by iteratively assigning each point of the distribution to the closest mean value and subsequently updating the mean based on all assigned points. This is a simplified version of the more general Expectation Maximization (EM) algorithm, where only the mean values of k Gaussians are estimated and all covariance matrices are fixed to the identity matrix. The dictionary is then defined by the k found mean values, their indices $1 \dots k$ to be more precise, and vector quantization of a descriptor is equivalent to nearest neighbor search. While k-Means is very popular and accurate in terms of matching the closest visual word,

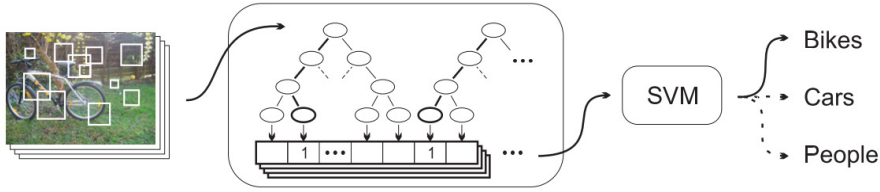


Figure 2.5: Concept of Extremely Randomized Clustering (ERC) forests applied to image classification. Feature descriptors are extracted sparsely at interesting keypoints of different scales. Every feature is then vector quantized by traversing it through the tree. The distribution of observed leaf node indices is then used as image-level descriptor and classified with a Support Vector Machine (SVM). Image courtesy of Moosmann et al. (2008).

it is rather inefficient, as high-dimensional nearest neighbor search against hundreds or thousands of candidates is very time consuming, with the runtime complexity being $O(kD)$, where D is the feature dimensionality. As a solution to this problem, Nister and Stewenius (2006) introduced vocabulary trees, which are an approximation to k-Means and are realized as k-Means hierarchy. They provide a good trade-off between runtime and matching accuracy: With a tree-structured hierarchy of l levels, and a small number of m cluster centers per level, the complexity can be reduced to $O(lmD)$, where $l \approx \log k$.

DECISION TREES Moosmann et al. (2008) introduced Extremely Randomized Clustering (ERC) forests as a means of rapidly quantizing feature descriptor vectors to visual words. Similar to the hierarchical k-Means-based quantization of Nister and Stewenius (2006), ERC utilize a tree-structure to yield efficient inference. However, the decision to which node in the tree a descriptor traverses next does not involve a full D -dimensional nearest neighbor search against m candidates, but only a single scalar threshold comparison. This further reduced runtime complexity to $O(\log k)$. In addition to more efficient inference, ERC forests have also shown better task-specific quantization performance than k-Means, because the trees can be trained in a supervised manner, compared to unsupervised training in case of k-Means. Figure 2.5 depicts the concept of ERC forests applied to image classification. We will adopt this encoding principle later in Section 5.1.2, where we provide more technical details of the method and discuss how it can be applied to scene labeling in our context.

2.3.2 Pyramidal Encoding

While the orderless nature of the BOF model works well in many cases, introducing more spatial structure can sometimes improve match-

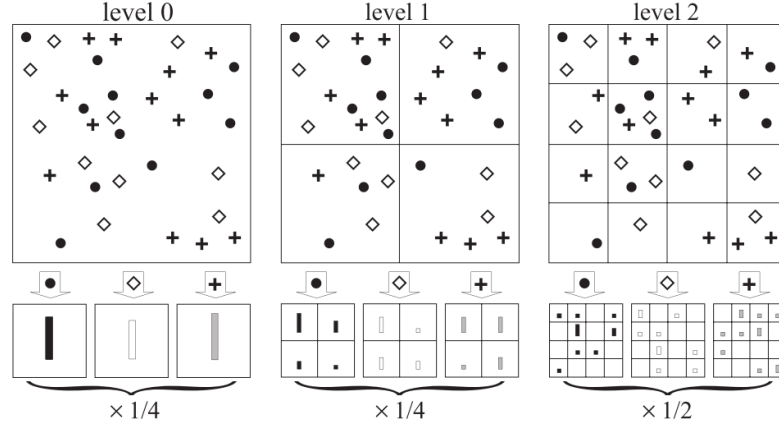


Figure 2.6: Concept of spatial pyramid matching. To consider the position of local features in the image, histograms are computed over recursively smaller image partitions. Figure taken from [Lazebnik et al. \(2006\)](#).

ing and recognition performance. [Grauman and Darrell \(2005\)](#) introduced the *pyramid match kernel*, which essentially builds a pyramid in feature space and weights matches of two sets of features according to the level they occur on, where matches on a finer level are weighted stronger. This concept enabled much faster matching with an accuracy comparable to other methods at that time, but did not take spatial image structure into account. In order to change that, [Lazebnik et al. \(2006\)](#) introduced *spatial pyramid matching* as an alternative to ([Grauman and Darrell, 2005](#)), where the pyramid is not build in feature space but in 2D image space. This is done by recursively subdividing an image into its four quadrants and computing a separate BOF histogram for each such quadrant, *c.f.* Figure 2.6. While the histogram response of two images might be similar or even identical on the first level, every subsequent level adds finer details of the spatial feature distribution. The final histogram is then obtained by concatenating all histograms of all levels, resulting in a very high-dimensional but sparse image descriptor. Later in this dissertation we introduce *height pooling* (*c.f.* Section 5.1.3), which is an extension of this principle that takes into account depth information.

2.4 SCENE LABELING

The approaches introduced in Section 2.1 build a camera-based environment model by reasoning from motion and depth cues and do not take into account appearance information. Also, the parameters of these models are tuned manually. Scene labeling on the other hand aims to infer the semantic category of each pixel in an image, typically by extracting and classifying color and texture features that encode the appearance of an image region. The parameters to find a good

*On the first level,
the model is still
orderless.*

*The terms scene
labeling and
semantic
segmentation are
used
interchangeably in
this dissertation.*

classification model are typically learned from data by means of machine learning. In the following, we discuss an excerpt of the large body of scene labeling literature. We roughly separate existing work as follows: (1) In Section 2.4.1, we discuss various scene labeling methods in general, including approaches that make use of depth information. On the highest level, we distinguish between pixel-level approaches and region-level approaches, which we aim to unify in this dissertation. (2) In Section 2.4.2, we introduce scene labeling methods that perform temporal reasoning for semantic video segmentation, which is another problem we focus on in this dissertation. (3) Finally, Section 2.4.3 covers a more detailed discussion of scene labeling methods that will be used as reference later in this work.

2.4.1 *Pixel- and Region-level Scene Labeling*

Over the last decade, there have been different methodological attempts to solve the scene labeling problem, starting with dense pixel-level classification, then superpixel classification, and finally classification of larger bottom-up region proposals. Following the latter two approaches mainly speeds up the process as fewer entities have to be classified, but introduces an intermediate step to extract superpixels or region proposals that in turn can also be costly or introduce errors. One of the goals of this dissertation is to combine pixel-level and region-level approaches within a single architecture to retain the precision of pixel-level models and the efficiency of region-level models (*c.f.* Chapter 6). In the following, we discuss related scene labeling models according to this separation.

In the realm of dense pixel classification, autocontext models have been proposed to integrate semantic context in labeling (Fröhlich et al., 2012; Shotton et al., 2008; Tu and Bai, 2009). The term autocontext is used if a sequence of classifiers is applied and the output of the previous classifier is used as input for the next classifier. In this way, every stage has more contextual information available as input. This concept is used in a variety of problems outside of semantic segmentation, most notably for body pose estimation, where *e.g.* a nearby limb is a strong indicator for hands or feet. Although computationally expensive compared to region-based approaches, real-time capable implementations of dense pixel-level semantic segmentation have been proposed *e.g.* by Costea and Nedeveschi (2014) and Shotton et al. (2009).

Region-based schemes have been applied in many approaches in order to obtain more discriminative features and to allow efficient inference. While fine-grained superpixels have been used as regions (Fulkerson et al., 2009; Mičušík, 2009; Zhang et al., 2010), recent work shows impressive performance using segmentation trees that contain larger region hypotheses (Arbeláez et al., 2012; Carreira et al.,

2012). Fraundorfer et al. (2010) use stereo information to rectify image patches, which results in viewpoint invariant patches (VIP) that are quantized together with SIFT (Lowe, 2004). Furthermore, features obtained from dense depth maps have also been quantized in the context of gesture recognition, e.g. by Hernández-Vela et al. (2012).

In an early work of Hoiem et al. (2007) the surface layout is extracted from monocular images using features such as location in the image, shape of superpixels, color, filter bank responses for texture and vanishing lines. In case depth information is available, rich additional features can be extracted, to capture the 3D structure of the scene. A repeatedly used approach is to extract surface normals and height above the ground plane for each pixel to support the segmentation task (Gupta et al., 2013; Zhang et al., 2010). Sparse point clouds obtained from Structure From Motion (SFM) have been used for image segmentation both in isolation (Brostow et al., 2008) and in combination with image-based cues, such as appearance and color (Mičušík, 2009; Sturgess et al., 2009). The results indicate that the height of feature points relative to the camera is an informative cue. Ladický et al. (2010) propose a joint formulation of semantic segmentation and dense stereo reconstruction within a pixel-wise conditional random field framework. Co-dependencies and interactions between segmentation and stereo reconstruction help to increase segmentation performance. We will discuss this work in more detail later in Section 2.4.3, as it is one of the reference methods we compare our approach against.

More recently, deep convolutional neural networks (CNNs) have become a popular choice to solve the scene labeling problem, due to their ability to encode complex image features and spatial relationships. Due to the typical “bottleneck” architecture of deep scene labeling models, where higher-level features are composed of a weighted combination of lower-level features, there is no need to differentiate between pixel-level and region-level models that encode local structure and global context respectively. Both aspects are encoded jointly in the model. Another advantage of deep networks is the possibility to train them end-to-end, i.e. to jointly optimize feature encoding and classification in one holistic model. Notable instantiations of deep convolutional networks for scene labeling include the Fully Convolutional Network (FCN) approach by Long et al. (2015) and SegNet by Badrinarayanan et al. (2017), which have a similar feature encoder stage but diverge at the point where the embedded feature representation is decoded back to original image resolution.

2.4.2 Temporally Consistent Scene Labeling

On a high level, existing work can be distinguished into *offline* and *online* methods. Offline or batch methods, e.g. (Grundmann et al., 2010; Kundu et al., 2016), have the potential to yield best possible segmenta-

tion performance, as they can access all available information from all time steps during inference. However, as they are not causal, they cannot be applied to streaming video analysis, which is a requirement for many applications, such as mobile robotics. In contrast, online methods perform *forward inference*, *i.e.* they only take into account observations from previous time steps. A closer look reveals fundamental differences, mainly separating *recursive* models (Erbs et al., 2012; Wojek and Schiele, 2008; Wojek et al., 2013) from models considering longer time history (Ellis and Zografos, 2012; Floros and Leibe, 2012; de Nijs et al., 2012). The latter also include models performing inference in a 3D graphical model (space and time) over a stack of several frames. Furthermore, the position in the processing pipeline and the level of abstraction on which temporal consistency is enforced has several important implications. For example, low-level motion segmentation (detection-by-tracking) methods, such as presented by Ellis and Zografos (2012) or Ochs and Brox (2011), can provide temporally stable proposal regions as input for semantic labeling but require prominent motion of an object in the image for proper detection. Miksik et al. (2013) propose a post-processing algorithm for causal temporal smoothing of frame-by-frame segmentation results, where temporal contributions are weighted according to a pixel-wise similarity measure, which requires dense optical flow.

As discussed in Section 2.4.1, prevalent consensus has emerged that increasing the level of abstraction from pixels to superpixels or larger image regions allows for richer models and more efficient inference. In fact, most state-of-the-art methods rely on superpixels, *e.g.* (Arbeláez et al., 2012; Carreira et al., 2012). However, as superpixels are typically built in a bottom-up fashion, their boundaries often fluctuate when applied to consecutive frames in a video sequence. The difficulty of registering and aligning superpixels over time has been addressed by Couprie et al. (2013) and Vazquez-Reina et al. (2010). Alternatively, using the warped previous segmentation as initialization for superpixels in the current frame is done by Abramov et al. (2012) and Mester et al. (2011). In (Tang et al., 2013), spatio-temporal segments are extracted from video data and are subsequently ranked according to weakly labeled categories provided with the video. However, even with perfect temporal registration of superpixels and object shapes, the semantic label decision can still be incorrect, mainly due to temporal noise in the classification results, *c.f.* Tighe and Lazebnik (2010).

2.4.3 Scene Labeling Reference Methods

Later in this dissertation, we will compare our results against other methods from the literature that are either technically related, make use of stereo depth information or deliver state-of-the-art scene label-

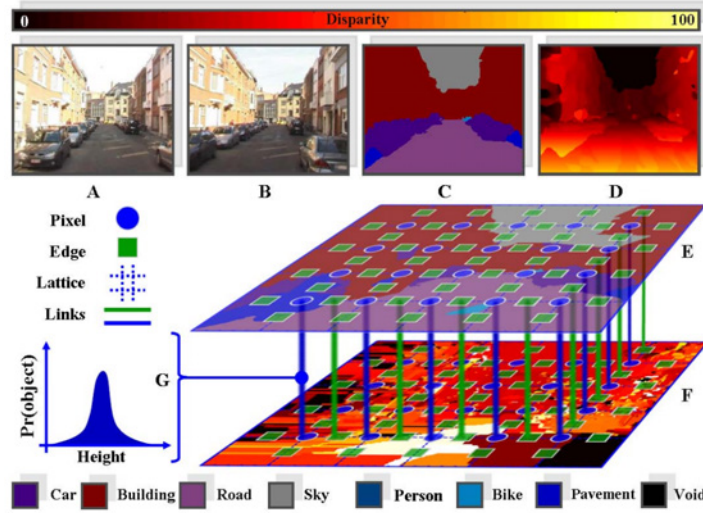


Figure 2.7: Overview of the ALE labeling method. Given a pair of stereo images, semantic object categories and stereo disparity values are estimated jointly within a dense CRF framework. Image taken from Ladický et al. (2010).

ing performance on the datasets we use for evaluation. This section introduces those methods in more detail.

ALE Ladický et al. (2010) present a method to jointly estimate semantic object categories and dense disparity maps from a pair or stereo images. The main rationale followed in this work is that estimating categories and depth jointly improves the result of both modalities, as there is a strong correlation between them, *i.e.* knowing that a pixel is part of the road increases the chance of measuring a constant disparity gradient along the vertical image dimension and vice versa. Also, sudden changes in depth often coincide with a transition of the class label. Their model is formulated as a dense Conditional Random Field (CRF), where each random variable can take a label from the product space of categories and disparity values, *c.f.* Figure 2.7. As the number of possible output configurations is very large, they employ an approximate inference using graph cut-based move making algorithms. Code for this method is published as part of the Automatic Labeling Environment (ALE), a software framework provided by the Oxford Brookes Vision Group¹. This model delivers high accuracy but requires by far the most computation time compared to all other discussed methods.

DARWIN Gould (2012) introduces DARWIN, a framework for computer vision and machine learning research that contains a boosted decision tree classifier to predict pixel-level unary scores (Gould et al., 2009), as well as a dense CRF model with contrast-sensitive pairwise

¹ <http://www.robots.ox.ac.uk/~phst/ale.htm>

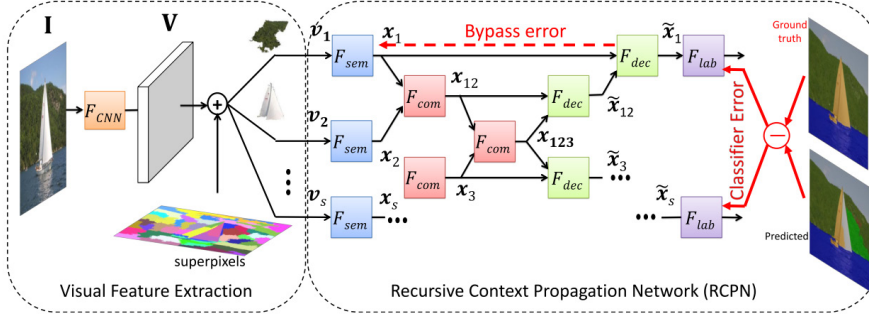


Figure 2.8: Overview of the RCPN labeling method. Semantic context is propagated recursively within a superpixel tree by means of neural networks that combine and decombine features. Image taken from Sharma et al. (2015).

terms, which is a common technique to favor a class transitions if image intensities change. We report numbers generated with this framework as baseline for unary pixel-level classification and standard dense CRF regularization that does not take into account depth information. The source code of the DARWIN framework is provided publicly².

PN-RCPN Sharma et al. (2015) present a scene labeling method based on recursive context propagation networks (RCPN). The key idea of this approach is to distribute semantic contextual information within trees of superpixels. Starting from the root node that represents the entire image, semantic context is recursively propagated from parent nodes to child nodes, so that all superpixels (the leaves in the tree) are expected to contain global contextual information. To this end, different neural networks are learned to (1) map local image features to semantic features, (2) combine child node features to parent node features, (3) decombine parent node features to child node features and finally (4) map context-enhanced features to labels, *c.f.* Figure 2.8. Local image features are extracted with the multi-scale CNN of (Farabet et al., 2013) and averaged over superpixels. PN stands for *pure node*, which denotes the best performing version of the method. This method does not use depth information, but delivers state-of-the-art performance on one of the datasets used for evaluation.

LAYERED INTERPRETATION Liu et al. (2015) propose a model for jointly estimating semantic labels and disparity maps under strong geometric scene constraints. The concept is in fact highly related to the Stixel idea, but imposes stronger model constraints. Each column is divided into at most four horizontal layers and all columns are solved separately. The order of labels in the layers is constraint according to Figure 2.9 and there is a fixed depth ordering from close to

² <http://drwn.anu.edu.au/>

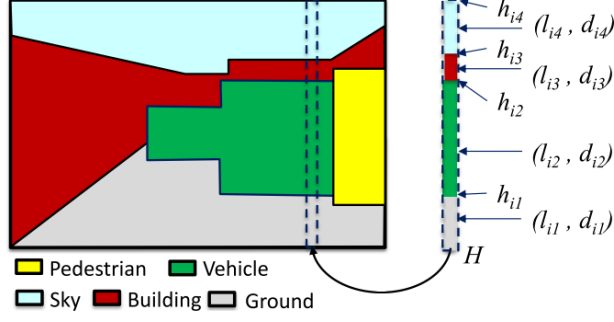


Figure 2.9: Scene model of the Layered Interpretation method. Each column can have one to four layers, where the order of labels is fixed. Image taken from Liu et al. (2015).

far when moving from the bottom to the top of the image. These constraints are formulated as an energy minimization problem that has only five free variables per column and can be solved with dynamic programming. As input, this method uses depth and appearance cues. Depth cues come from a cost volume that contains the intensity difference for every possible disparity value, which is a common choice for many stereo algorithms. For appearance information, they use the RCPN method of Sharma et al. (2015), so the proposed *Layered Interpretation* model can actually be seen as an extension of RCPN with more scene constraints and use of depth information. Their approach is implemented on a GPU and provides very competitive results at low computation time on the tested datasets. It is a good example of a method that achieves high performance by imposing strong scene constraints.

SEMANTIC TEXTON FORESTS Shotton et al. (2008) propose the semantic texton forest concept for image segmentation, which consists of two major parts: The first part is a pixel-level RDF, the Semantic Texton Forest (STF), which operates on small rectangular image patches and is used to extract dense texton maps. For every tree in the STF, the texton map encodes the ID of the leaf node each pixel falls into. This is essentially the ERC feature encoding concept of Moosmann et al. (2008) (c.f. Section 2.3.1), but applied directly on image pixels and computed densely, compared to encoding descriptors extracted at sparse keypoints. In addition to texton maps, the STF also infers an initial pixel-level labeling that is used later as a region-level semantic prior. Texton maps and semantic prior are visualized in Figure 2.10.

The second part of the concept is the Bag-of-Semantic-Textons (BOST) histogram, i.e. a histogram of textons, accumulated over an image region and augmented by attaching the average class label distribution from the STF leaf nodes as a region-level semantic prior. This BOST histogram then constitutes the final region-level descriptor, which is shown in Figure 2.11. To infer the semantic segmentation of an image,

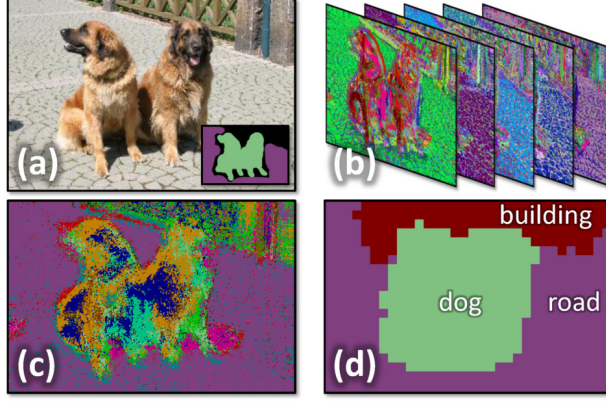


Figure 2.10: Overview of the semantic texton forest approach. (a) Input image and ground truth, (b) textons maps for five trees, (c) semantic prior computed with the STF, and (d) final result. Image taken from Shotton et al. (2008).

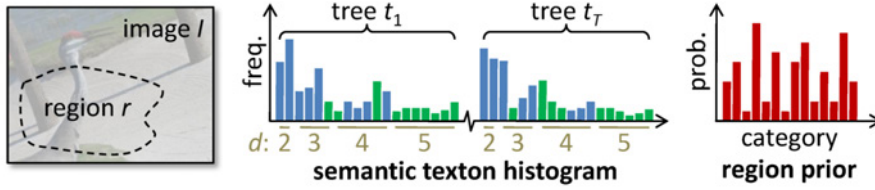


Figure 2.11: Bag-of-Semantic-Textons (BOST) descriptor to encode an image region r . In contrast to the ERC concept of Moosmann et al. (2008), intermediate nodes in the tree also contribute to the histogram to additionally capture the tree structure. Image taken from Shotton et al. (2008).

the TextonBoost approach (Shotton et al., 2006) is adapted by replacing the boosting classifier with an RBF and replacing a histogram of pre-defined quantized textons with the learned BOST descriptor. The TextonBoost approach analyses multiple informative image regions of varying size and shape around a pixel of interest, where the maximum distance of these regions to the pixel can be rather large (up to half the image size), so that semantic long-range context can be learned.

CONTENTS

3.1	Datasets for Outdoor Urban Scene Labeling . . .	32
3.1.1	CamVid	32
3.1.2	Leuven	33
3.1.3	KITTI	33
3.1.4	Discussion	34
3.2	The Daimler Urban Segmentation Dataset	35
3.2.1	General Description	35
3.2.2	Evaluation Protocol	38
3.2.3	Dataset Statistics	41
3.3	The Static Estimator	44

In recent years, significant progress in computer vision has been made possible with the help of machine learning techniques. The goal of machine learning is to automatically learn the free parameters of a model, so that the model is able to generalize from provided data samples and infer the desired outcome for previously unseen input. Datasets are a central enabler for this technology as they are a prerequisite for learning the free parameters. At the same time, datasets are also crucial to quantify how well a certain problem can be solved and to compare different solutions against each other. In fact, most datasets do not only contain the actual input data for an algorithm, but also some form of reference data that defines the desired outcome of an algorithm. This reference is referred to as *ground truth*, a term we will use frequently throughout the remainder of this dissertation. Generating ground truth is often a tedious task, as it typically requires human annotators to manually generate the desired outcome for a given input. In the setting of outdoor semantic scene labeling, this means the annotator is presented with an image of an outdoor scene and has to define the target class label for each individual pixel in the image.

In a supervised setting, where the desired outcome is available during training, ground truth data is required for both training *and* evaluation. It is thus important to have a well-defined separation of the data into disjoint parts for training and evaluation. Otherwise, it is utterly difficult to obtain an unbiased estimate of the true performance of a method. The reason for this is *overfitting*, which happens when an algorithm overly specializes to the samples it is trained on and

In contrast, unsupervised methods try to discover patterns from data where no ground truth is available.

hence loses the capability to generalize to unseen data. Without a separation into training and testing parts, this is very hard to detect.

Furthermore, not only annotated data is important to compare different methods against each other. The definition and widespread use of meaningful evaluation metrics is key to enabling a fair and flawless comparison. Unfortunately, there is almost never a single measure that can capture all relevant aspects of a method properly, as demands on an algorithm can differ significantly depending on the use case.

In this chapter, relevant existing datasets for scene labeling of urban streets are reviewed and discussed with regard to their settings and properties. Subsequently, a novel dataset for urban scene labeling is introduced that aims to avoid the shortcomings of previous datasets. Furthermore, different evaluation metrics for scene labeling are reviewed and related to each other later in this chapter.

3.1 DATASETS FOR OUTDOOR URBAN SCENE LABELING

In the following, three related datasets are discussed that have been used in the context of semantic labeling of outdoor street scenes: *CamVid*, *Leuven*, and *KITTI*. While there is a large body of literature reporting numbers on these datasets, they are all in all not well suited for our task, either in terms of size and resolution or because they do not provide all input modalities that should be explored in the scope of this dissertation. As a result of this finding, a novel dataset will be proposed subsequently that aims to fill this gap.

3.1.1 *CamVid*

The *CamVid* dataset was introduced by [Brostow et al. \(2009\)](#). They were the first to propose a large dataset to evaluate semantic segmentation in the context of outdoor urban street scenes. In contrast to previous datasets, they made the move from still images to video sequences, which are recorded using a vehicle-borne camera. The authors anticipated that this step allows the development of more elaborate algorithms that take into account temporal context. From the 10 minutes of video material, 701 frames come with manual annotations into 32 semantic classes. An example image of the *CamVid* dataset with corresponding semantic annotations is given in Figure 3.1.

Unfortunately, the *CamVid* dataset is not recorded with stereo cameras, so that depth information can only be extracted from motion analysis. Structure From Motion (SFM) methods such as used by [Brostow et al. \(2008\)](#) and [Sturges et al. \(2009\)](#) allow to estimate depth from a video stream using a single camera, but the reconstruction of these approaches is quite sparse and the quality is still not comparable to a well-calibrated stereo camera setup. This is mainly due

to the required 2D matching of correspondences, whereas the stereo case only involves 1D matching along the epipolar line (Hartley and Zisserman, 2004). Furthermore, SFM only works in a stationary environment, where the camera is the only moving object. For in-vehicle applications, where not only the observing camera but also other traffic participants are potentially moving, this is arguably a strong limitation.

3.1.2 *Leuven*

The *Leuven* stereo dataset was originally introduced by Cornelis et al. (2008) and later extended with semantic class label and depth annotations by Ladický et al. (2010). The dataset contains 1175 stereo image pairs, where the two cameras are mounted on top of a moving vehicle and are 150 cm apart from each other. The images are recorded as a video stream over a distance of about 500 meters in an urban neighborhood and it was the first scene labeling dataset with video and stereo information available. Unfortunately it suffers from severe problems that make it impractical to use from today's perspective: the images have a resolution of only 360×288 , poor illumination conditions and motion blur. Semantic class label annotations are available for only 70 images, where 50 images are used for training and the remaining 20 for testing. In light of this fact, it is questionable whether the results would actually provide strong statistical evidence of a method's performance. Furthermore, almost no pedestrians are visible at sufficiently high resolution and the overall scene is rather static. Figure 3.1 also provides an example of this data.

3.1.3 *KITTI*

The *KITTI* Vision Benchmark Suite (Geiger et al., 2012) is a dataset collaboratively built by the Karlsruhe Institute of Technology and the Toyota Technological Institute in Chicago (KITTI). It was introduced to assess and support research on vision algorithms for autonomous vehicles. The dataset includes recordings in urban, rural and highway scenarios and provides data from stereo cameras and a Velodyne 360° laser scanner, both mounted on top of a mid-size car. Based on the recorded data, the authors compiled different challenges for problems such as stereo vision, optical flow, visual odometry and object detection and provide rankings of the best performing algorithms on their website¹.

While the *KITTI* data itself is very suitable for our setting of outdoor urban scene labeling, there is no official scene labeling benchmark or ground truth data available at the time of this writing. However, recently several authors have manually annotated different parts

¹ www.cvlibs.net/datasets/kitti/

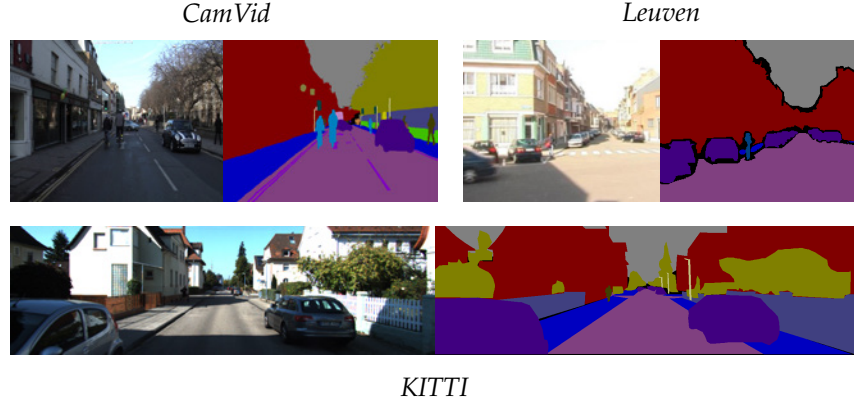


Figure 3.1: Example images taken from three related datasets: *CamVid* (Brostow et al., 2009), *Leuven* (Ladický et al., 2010), and *KITTI* (Geiger et al., 2012) with annotations from Ros et al. (2015). Corresponding semantic class annotations are provided on the right of each image, where colors encode different class labels.

of the *KITTI* data with pixel-level semantic labels (He and Upcroft, 2013; Ladický et al., 2014; Ros et al., 2015; Sengupta et al., 2013; Xu et al., 2013). Despite the varying objectives of the individual contributions and correspondingly varying annotation protocols, a subset of overlapping classes can be found across all the different parts.

We will make use of this data later in this dissertation by mapping all of the provided ground truth labels to a subset of classes relevant for our task. In doing so, we obtain a total of 427 images with manual annotations. To avoid overlap between training and testing data, we test on all 216 annotated images taken from the visual odometry part of the *KITTI* data (Ros et al., 2015; Sengupta et al., 2013) and use the remaining 211 images (He and Upcroft, 2013; Ladický et al., 2014; Xu et al., 2013) for training. In Figure 3.1 we show a *KITTI* image with annotations taken from Ros et al. (2015).

3.1.4 Discussion

The datasets presented in this section are all interesting for our general setup, but none of them provides all the information we would like to use in this dissertation. The *CamVid* dataset does not provide stereo imagery and the *Leuven* dataset is too small in terms resolution and size. The presented datasets are briefly summarized with their key properties in Table 3.1. From this listing it can be seen again that the collected *KITTI* annotations are in fact most interesting in the context of this dissertation, as they provide all necessary data we would like to use. However, the annotations have been published just recently and do not constitute an official benchmark as a whole. Nevertheless, we will repeatedly use this data as a second benchmark, to

Dataset	<i>CamVid</i>	<i>Leuven</i>	<i>KITTI</i>
Resolution	960×720	360×288	1240×370
# Images	701	70	427
# Classes	32	7	12
Color	yes	yes	yes
Video	yes	yes	yes
Stereo	no	yes	yes

Table 3.1: Overview of the key characteristics of the discussed scene labeling datasets, including resolution, number of annotated images, and number of classes.

demonstrate that our proposed methods are able to generalize across different datasets.

3.2 THE DAIMLER URBAN SEGMENTATION DATASET

In this section, we introduce the Daimler Urban Segmentation dataset, a novel dataset for semantic labeling of outdoor urban street scenes. Given that most popular datasets for semantic segmentation do not provide stereo images or are too small in terms of resolution and number of annotated frames, we present a challenging stereo vision dataset captured from a moving vehicle in complex urban traffic with manually annotated ground truth. With our dataset, we aim to provide data that allows to leverage motion and depth cues alike and exceeds previously used datasets in terms of resolution and size. In this section, we will discuss the general setting of our dataset, introduce the recording and annotation protocol and provide some general statistics of the data to provide better insight. We will use this dataset as the main source to train and evaluate the methods proposed throughout this dissertation and abbreviate it with *DUS*.

Our dataset is publicly available at www.6d-vision.com/scene-labeling/.

3.2.1 General Description

Our Daimler Urban Segmentation dataset contains five video snippets with 1000 consecutive stereo image pairs each. The videos are recorded using industrial monochrome cameras with a spatial resolution of 1024×440 pixels and 12 bit dynamic range, a setting often found in current industrial in-vehicle vision applications. The cameras deliver a frame rate of 25 Frames Per Second (FPS), so that each snippet covers a time span of 40 seconds.

The five snippets we selected for human annotation are taken from longer video sequences recorded in the context of the Daimler internal *Bertha* research project that was officially introduced to the public

in August 2013. The goal of this project was to build a vehicle with close-to-production sensors that is capable of driving autonomously in rural and urban traffic (Ziegler et al., 2014). On the sensor side, computer vision played an important role and covered tasks such as recognition of lanes and curbs, localization, object detection, tracking, motion segmentation and traffic light recognition (Franke et al., 2013). It is therefore easy to argue that the recorded video material is an interesting testbed for the design of novel algorithms as well, as the data was actually used to realize a successful autonomous vehicle.

The name of the project derives from the eponymous *Bertha Benz Memorial Route* (Figure 3.2) that has been chosen for testing and demonstration. Located in Baden-Württemberg, Germany, the route stretches over roughly 100 kilometers of rural roads, narrow villages and inner-city streets from Mannheim to Pforzheim, passing the city of Heidelberg and the Bruchsal Castle. From the video material recorded along this route, we chose the most



Figure 3.2: Bertha Benz Memorial Route with selected points of interest. Courtesy of Ziegler.

interesting and dynamic inner-city parts that contain many pedestrians and vehicles. For both of the two classes, heavy occlusion is present in the images, with crowds of pedestrians crossing the street and parking cars mutually occluding each other. To provide better insight into our dataset, Figure 3.3 shows some example images together with the corresponding disparity maps and semantic annotations.

ANNOTATIONS For the given data, we identified seven object classes of interest: vehicles, pedestrians, bicycles and cyclists, drivable ground surface, buildings and sky. While this set of labels seems rather small compared to other related datasets such as CamVid with its 32 classes, we anticipate that a small number of classes with more per-class examples offers stronger statistical evidence when reporting numbers compared to many classes with only few instances per class present in the data.

From the 5000 stereo image pairs in our dataset, we manually annotated 500 images with pixel-level semantic labels. We followed an equidistant annotation protocol, *i.e.* labels exist for every tenth frame of our video data, starting at frame 10 and ending at frame 1000 of each sequence. We choose this protocol in favor of methods for online

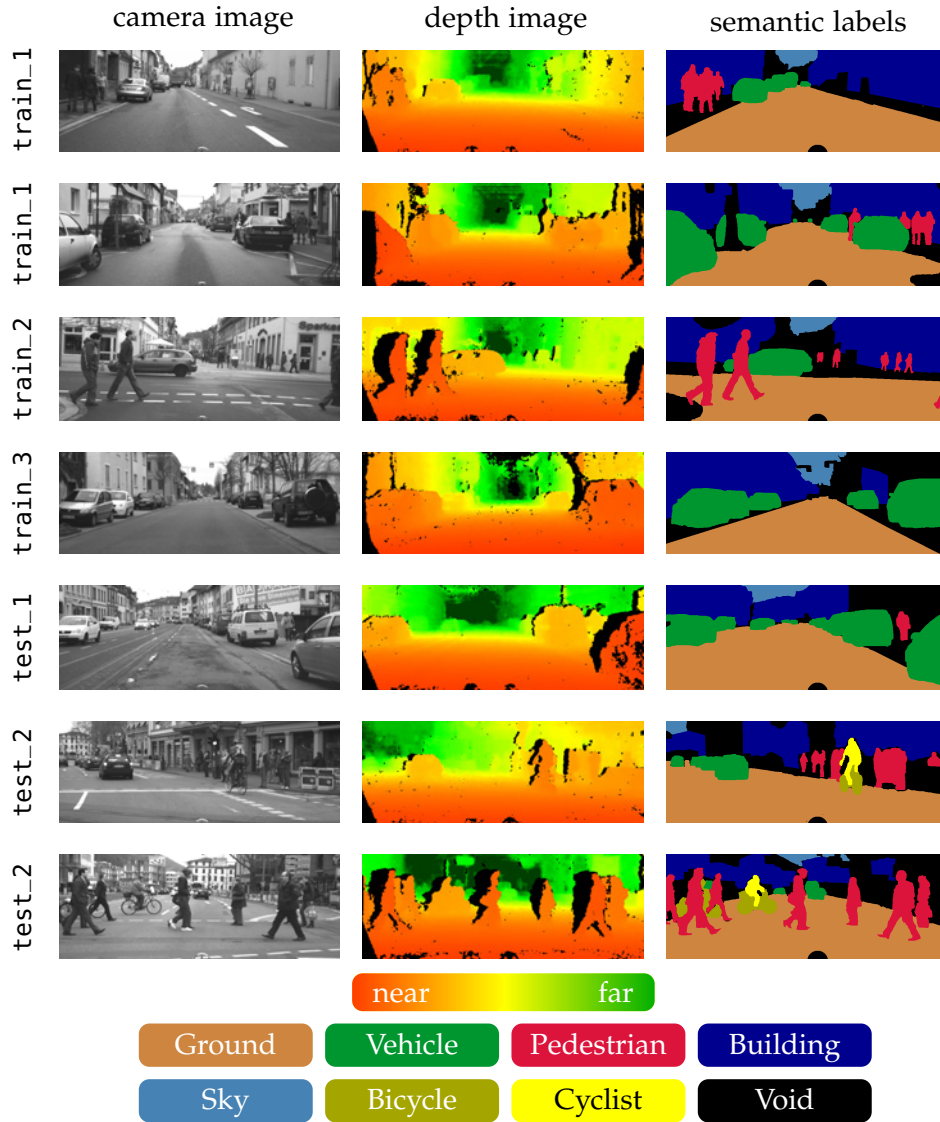


Figure 3.3: Examples taken from our dataset. The grayscale camera image is shown on the left, with the corresponding disparity map in the center and the pixel-level semantic class annotations on the right. The colors in the disparity maps encode near pixels in red and far away pixels in green. Black areas indicate regions where no matching is possible. Colors in the annotation images indicate the semantic class label, *i.e.* green for vehicles and blue for building. Black regions do not belong to any defined class label or are very cluttered and therefore left unlabeled. In total we provide annotations for seven class labels: ground, building, sky, vehicle, pedestrian, bicycle and cyclist. It can be seen that our images contain many articulated pedestrians, which makes precise segmentation very challenging. The name of the source sequence is given on the left of each image.

streaming analysis. In this way, there is a fixed time window in the beginning of each sequence and between annotated frames to let an algorithm leverage temporal cues so as to increase its certainty before the inferred result counts in the evaluation.

The tool we developed to generate the pixel-level labels provides several features to speed up the labeling effort, including drawing freely with a variable sized brush and creating polygons that close automatically. What we found most useful however was an additional mode that finds the shortest path along strong image gradients from the last click to the current mouse pointer position. This feature allowed to rapidly generate highly precise object contours with only a few clicks and was used intensively throughout our annotation process.

STEREO SETUP As pointed out earlier, one of the central reasons for creating this dataset is to provide additional depth information from a stereo camera setup. We use a stereo rig with a baseline of 35 cm and a field of view of 45° that is mounted behind the windshield inside of our recording vehicle. Intrinsic and extrinsic calibration of the stereo rig has been conducted carefully, so that stereo matching errors arising from miscalibration can be factored out. Consequently, all images in our dataset are already rectified so that for a point in the left image, the corresponding point in the right image is always on the same row index. To avoid unwanted decalibration of the setup, the two cameras are rigidly connected using a solid metal tube. In addition to the rectified left and right camera images, we also augment the dataset with pre-computed dense disparity maps that we generated using a Daimler-internal implementation of the well-established [SGM](#) algorithm proposed by [Hirschmüller \(2008\)](#). The resulting depth maps contain sub-pixel accurate disparity estimates in the range from 0 px to 127 px and provide measurements for about 90% of all pixels. While depth estimation with stereo cameras typically fails in regions with low contrast, the [HDR](#) cameras used for the recordings effectively avoid this problem, so that the predominant cause for invalid measurements is not low contrast but stereo occlusion, *i.e.* a pixel is visible in the left image but occluded in the right image. We detect and mark these invalid measurements using a left-right consistency check. Since we are using the left image as the reference frame, the resulting invalid regions are always on the left side of objects in the scene and the effect is more pronounced for objects closer to the camera.

The consistency check verifies that left-to-right matching and reverse right-to-left matching are in agreement.

3.2.2 Evaluation Protocol

Out of our five video snippets, we selected two sequences solely for evaluation, while the remaining three can be used for training and cross-validation of learning-based methods. For ease of use we

named all five sequences directly according to their purpose: train_1, train_2, train_3, and test_1, test_2, *c.f.* Figure 3.3 (left column). We chose one of the most challenging snippets (test_2) for evaluation, as this part contains strong object occlusions and dynamic motion of the recording vehicle and other traffic participants. It includes a twisty road layout, overtaking maneuvers, stopping at a traffic light, and pedestrians waiting and crossing the street in front of the vehicle. To avoid overfitting to a certain location, the split was further chosen so that locations present in the evaluation part are never occurring in the training part. While the provided separation into training and testing parts is well motivated and meaningful from our perspective, it comes at the cost that bicycle and cyclist labels are under-represented in the training set. This renders it impractical to train a strong classifier for these classes. We therefore decided to map all label instances of bicycles and cyclists to the pedestrian class for training and evaluation, so that in all experiments throughout this dissertation, we only consider a segmentation problem into five classes: ground, building, sky, vehicle, and pedestrian.

Note that in our dataset bicycles almost never occur in isolation but always together with a cyclist, which makes our mapping more plausible.

METRICS As briefly mentioned earlier in this chapter, the definition of meaningful and well-suited evaluation metrics is an important aspect when introducing a novel dataset. In the following, we will therefore discuss existing measures to compare semantic segmentation methods and eventually introduce the evaluation protocol we define for our dataset.

Common measures to evaluate semantic segmentation performance are Global Precision (**GP**), Average Precision (**AP**) and the Intersection Over Union (**IOU**) metric from the **PASCAL VOC** challenge (Everingham et al., 2010). All these metrics have in common that they are derived from the pixel-level confusion matrix between the ground truth annotations and the inferred outcome of an algorithm. A confusion matrix is a square matrix of size $L \times L$, where L is the number of classes, *e.g.* five for our dataset. For each annotated pixel in the test data, an entry is added with the corresponding pair of ground truth label and inferred label. Consequently, an inferred segmentation result that is identical to the ground truth annotation produces a confusion matrix with entries only along the diagonal of the matrix, where ground truth label and inferred label are identical. The **GP** measures how many of all pixels globally have been classified correctly, *i.e.*

$$GP = \frac{1}{N} \sum_{i=1}^L c_{ii}, \quad N = \sum_{i=1}^L \sum_{j=1}^L c_{ij} \quad (3.1)$$

where c_{ij} are the entries in the confusion matrix and N is the sum of all entries. A disadvantage of **GP** is that it has a strong bias towards classes that cover a large portion of the image, such as ground surface and buildings, while classes with smaller entities barely contribute to

the final score. It is therefore relatively easy to obtain high scoring results, even if the majority of classes is not inferred accurately or not inferred at all. To account for class imbalance, [AP](#) measures the accuracy for each class independently and averages the resulting numbers over all classes to obtain a single score. The class-wise accuracy is always given relative to the ground truth, *i.e.* “how many of all actual vehicle pixels have been classified as vehicle?”. This formulation directly translates into the [AP](#) definition, with

$$AP = \frac{1}{L} \sum_{i=1}^L P_i, \quad P_i = \frac{c_{ii}}{N_i}, \quad N_i = \sum_{j=1}^L c_{ij}. \quad (3.2)$$

Evaluating scene labeling accuracy in this way gives a better balance between the classes of interest. However, there are still some special cases that can lead to counterintuitive results. Consider the case where a method simply classifies all pixels to belong to one class, *e.g.* pedestrian. The P_i score for the pedestrian class would then be 100%, as all pedestrian pixels in the ground truth have been inferred correctly but false positives are not taken into account. To overcome this issue as well, [Everingham et al. \(2010\)](#) proposed the [IOU](#) measure, which is defined for class i as the intersection over the union of ground truth and inferred labels:

$$IOU_i = \frac{TP_i}{TP_i + FP_i + FN_i}. \quad (3.3)$$

True positive pixels (TP) are correctly classified pixels, false positives are pixels where class i was inferred but it is a different class in the ground truth and false negatives (FN) are pixels with class i in the ground truth that were missed and falsely assigned to a different class, *i.e.*

$$TP_i = c_{ii}, \quad FP_i = \left(\sum_{j=1}^L c_{ij} \right) - c_{ii}, \quad FN_i = \left(\sum_{j=1}^L c_{ji} \right) - c_{ii}. \quad (3.4)$$

In set theory, the [IOU](#) is also called *Jaccard index* and measures the similarity of two sets. Same as with the [AP](#) measure, the class-specific IOU_i scores are averaged over all classes to obtain a single scalar value. Results for the different measures are shown for an artificial confusion matrix in Figure 3.4. From the given example it is easy to see that all measures are given in percent, where 100% is the best possible score. It is furthermore apparent that the [IOU](#) is always lower than [GP](#) and [AP](#). It should be briefly mentioned here that [Csurka et al. \(2013\)](#) proposed an alternative measure to evaluate semantic segmentation, arguing that a per-image evaluation and down-weighting the influence of images with very low scores would be closer to a human

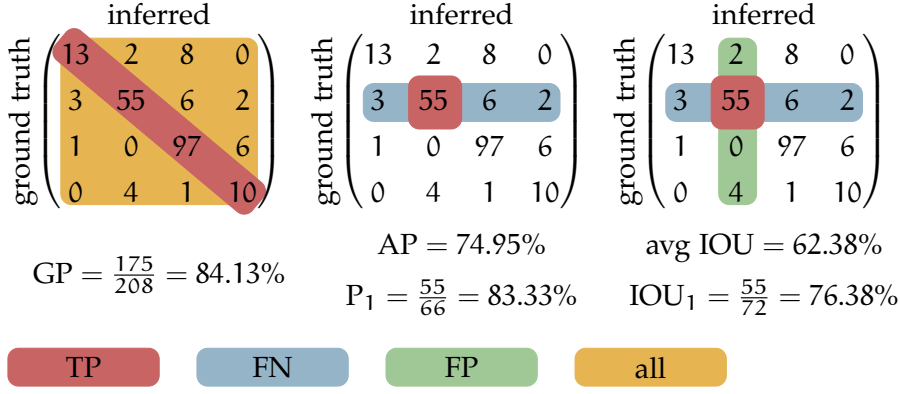


Figure 3.4: Artificial confusion matrix for a classification problem with four classes. Row indices denote the ground truth label while column indices correspond to the inferred label. We show three commonly used metrics to obtain a scalar performance measure from the given confusion matrix: Global Precision (GP), Average Precision (AP) and Intersection Over Union (IOU). If applicable, a single class-specific score is also given as example. Colors indicate the meaning of the contributing numbers.

quality estimate compared to IOU. We still use the IOU metric, as it is much more common in literature.

In the remainder of this dissertation, we will always report results using the IOU score, either for each class individually or averaged over all classes. In some cases, where we report only the average (Avg IOU), we will occasionally provide an additional average over the dynamic object classes only, *i.e.* vehicle and pedestrian (Dyn IOU). Doing so provides a better insight into the results, as these classes are very relevant for many applications and it is much harder to achieve high accuracy compared to ground, building or sky. In all cases, pixels that are not annotated in the ground truth do not contribute to the result.

3.2.3 Dataset Statistics

Before we start using our novel dataset, we will spend the next few pages to analyze and discuss a selection of interesting statistics that can be derived from the dataset. This information will help to provide a clearer picture of the challenges in our dataset and the aspects that set it apart from datasets existing in the literature. We will also see which parts are relatively easy and which parts hard to solve with this data.

To avoid confusion for practitioners working with our dataset, it should be noted that throughout this dissertation we are using the updated 2014 version of our dataset, which comes with small label corrections of the initially published version in (Scharwächter et al., 2013). Furthermore note again that we map all occurrences of cyclists and bicycles in our dataset to the pedestrian class label before work-

The main adjustments in the 2014 version are more accurate object boundaries.

		Ground	Vehicle	Ped.	Sky	Building
Pixel Share	Train	45.6	9.4	1.4	2.7	40.9
	Test	53.4	14.6	6.2	2.3	23.5
	Overall	48.8	11.6	3.4	2.5	33.7
Occurences	Train	319	685	542	363	732
	Test	217	702	475	235	497
	Overall	536	1387	1017	598	1229

Table 3.2: Distribution of annotations among the classes in our *DUS* dataset. We show the share of pixels [%] and the number of occurrences, separated by the parts for training and testing. An occurrence is defined by a closed image region of one particular label that covers more then 100 pixels. Unsurprisingly, the largest part is taken up by pixels covering the ground surface and buildings. In terms of occurrences however, there is more balance between the different classes.

ing with the data. The mapping from bicycle to pedestrian is not problematic in our case, as bicycles mostly occur alongside with cyclists. Hence, it is reasonable to define the combined object as a pedestrian.

In Table 3.2, we see how the annotations in our dataset are distributed among the classes. As expected for outdoor street scenes, pixels covering the ground surface and buildings prevail in this statistic. However, what is more important to point out here are the number of pixels covering vehicles and pedestrians. While 3.4% pixel share for pedestrians seems rather low at first sight, this number is relatively high compared to other scene labeling datasets. CamVid for instance has a pedestrian pixel share of only 0.67% (Brostow et al., 2009). The same applies for the vehicle class, where our dataset compares favorably in terms of labeled pixels and number of occurrences. This is an important factor, given that we have a strong focus on accurate detection and segmentation of these classes. We define a class occurrence as a closed image region of one particular label. To account for small annotation errors, we only count it as valid occurrence if the region covers more than 100 pixels. These occurrences however should not be confused with actual object instances in the scene. One occurrence can contain one or multiple occluded objects of the same class.

In a next experiment, we compute the image-level spatial prior distribution for each class in our dataset. The results are shown in Figure 3.5, where darker pixels indicate a higher chance that this pixel belongs to the respective class label. It can be seen that the results are in agreement with common sense, with ground covering the lower part of the image, sky covering the upper center, and buildings covering mostly background of the scenery. With respect to separability

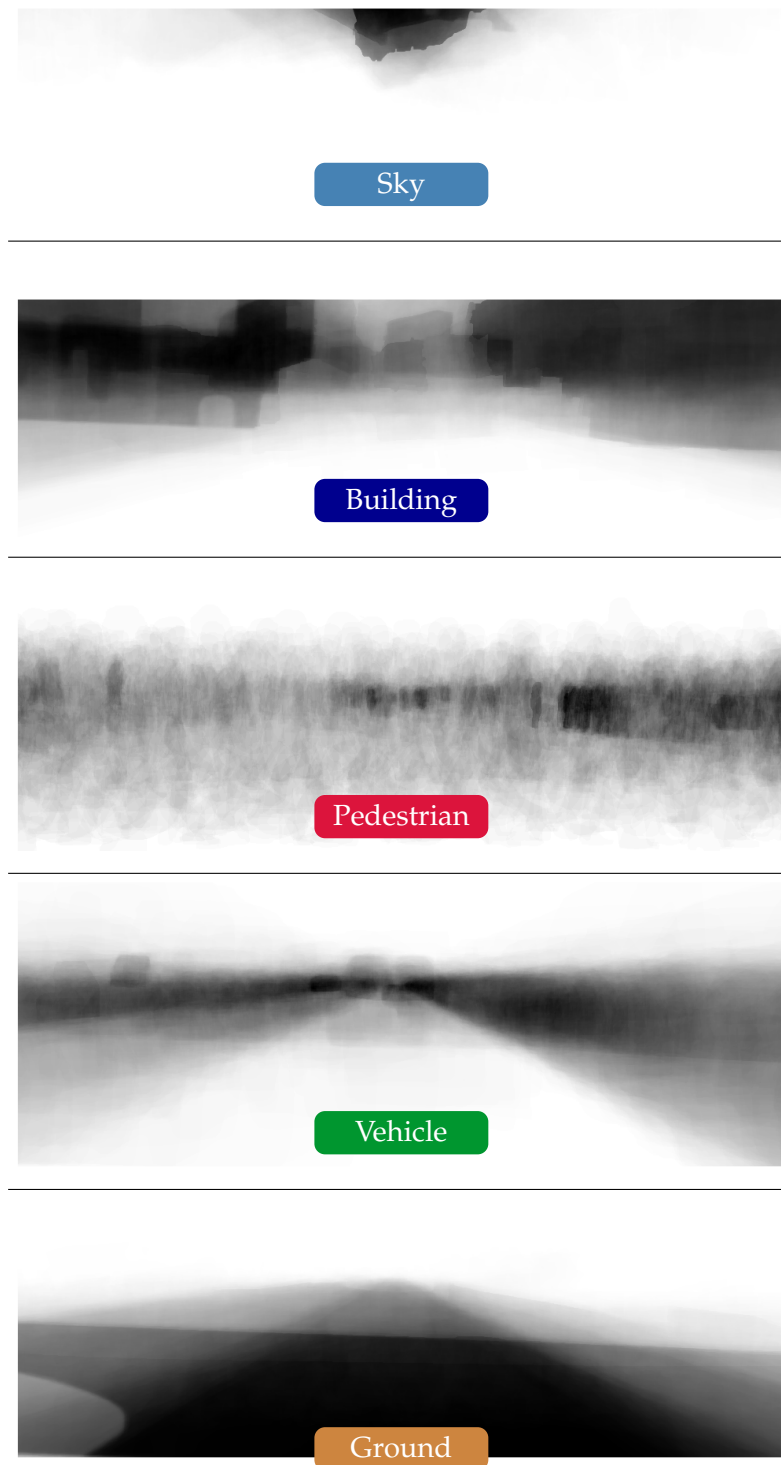


Figure 3.5: Pixel-level spatial location priors for the classes in our *DUS* dataset, computed over all 500 annotated images. The darker the image, the higher the chance that this pixel belongs to the respective class label.

of classes, there is a trend that the horizontal location in the image is rather uninformative, while a lot of structure is present in the vertical domain.

The vertical structure is even more pronounced when moving from the 2D image plane to 3D point locations. The depth information in our dataset can be used to transform each pixel with valid disparity measurement to its relative 3D location with respect to the camera. To visualize this information, we computed histograms over three different point transformations: height above the ground plane, distance to the camera in longitudinal direction and lateral distance to the camera center. The resulting distributions are shown in Figure 3.6. From the presented statistics we gain several interesting insights: (1) 3D height above the ground plane is a very informative w.r.t. class separability. In particular the coarse infrastructure labels ground, building, and sky have very smooth and distinct distributions in this domain. Vehicles and pedestrians on the other hand occur in the very same height range from 0 m to 2 m, so that other cues such as appearance in the image play a more important role. (2) Pixels in the sky are quite low in altitude, with the mean value roughly at just 8 meters above the ground plane. This is unexpected, but a typical artifact of the SGM stereo matching algorithm, which tends to bridge disparity values between well-textured objects like buildings, thereby assigning similar disparities to the in-between untextured sky area. (3) In contrast to 3D height above the ground, the longitudinal and lateral distance are quite noisy and the distributions of all classes strongly overlap. While the noise indicates a possible shortcoming in terms of dataset size, the lack of separability in these domains is in accordance with common sense. Relevant obstacles such as vehicles and pedestrians can generally occur at all lateral and longitudinal distances. Limiting the possible detection range based on prior observations therefore seems rather unnatural.

SGM tends to strongly smooth over untextured areas.

3.3 THE STATIC ESTIMATOR

In the previous section, we discussed different performance measures. However, numbers alone are not helpful without the ability to interpret them. To make sense of the numbers and support interpretability of the results provided later in this dissertation, we will now discuss a very primitive reference method based on the spatial location priors shown in Figure 3.5. The method simply always returns the class label with highest pixel prior probability in the training data. We call this method the *static estimator*, as it always returns the same result, irrespective of the actual input image. The idea behind this experiment is to get a first impression of the resulting scores. It should obviously not be misunderstood as a competitive method, but the results should mark the lower end of the interesting performance range.

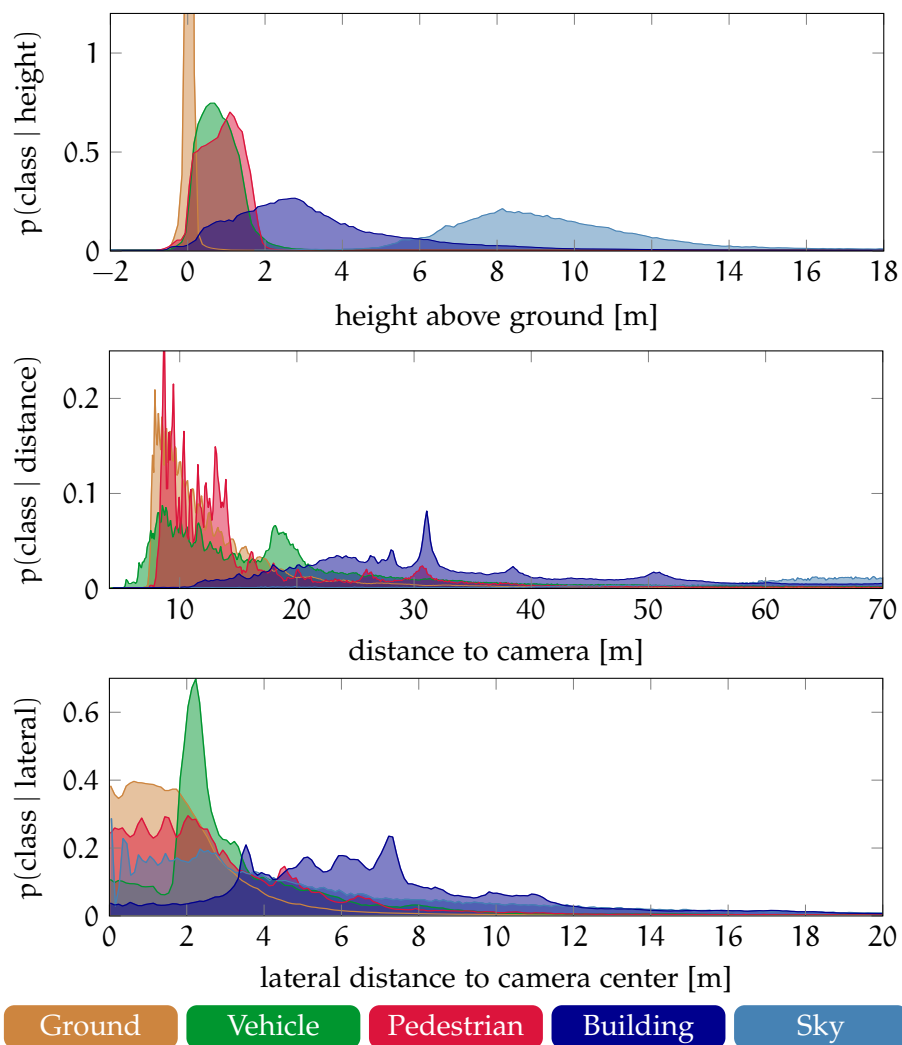


Figure 3.6: Class-specific distributions for three different 3D point projections. For each annotated pixel with valid stereo information, we transform it into height above the ground plane, distance to camera (longitudinal) and distance to camera center (lateral). Colors indicate the different class labels. It can be seen that height information is very smooth and classes are well-structured in this domain, while the other two transformations are quite noisy and classes strongly overlap.

	Gnd	Veh	Ped	Sky	Bld	Avg	Dyn
IOU	79.8	11.3	0.0	26.3	58.1	35.1	5.6

Table 3.3: Classification accuracy of the static estimator on the *DUS* dataset. The method delivers relatively high scores for ground and building already, due to the rather typical layout of outdoor street scenes. Vehicle performance on the other hand is very poor and pedestrians are not detected at all, *c.f.* Figure 3.7. *Dyn* denotes the average over the two classes Vehicle and Pedestrian only.



Figure 3.7: Best class label according to spatial location, computed using all training images in our dataset. Note that no pedestrians occur at all, as they are less frequent in the data.

The resulting **IOU** scores for this experiment on our *DUS* dataset are shown in Table 3.3, together with an illustration of the static inference outcome in Figure 3.7. As expected, the method gives rather low scores on average, in particular for the dynamic classes, *i.e.* vehicles and pedestrians. However, the ground surface is already detected quite well, due to the typical layout of outdoor street scenes and the fixed position and orientation of the installed camera.

The static estimator is also interesting to compare different datasets with respect to scenario variability. If the prior distribution alone already provides good scores, there is probably a strong bias towards a certain scenario in the dataset. Table 3.4 shows the static estimator results for all datasets discussed before. Note that the average performance of the static estimator is lowest on our *DUS* dataset, indicating that it has higher variability than the other discussed datasets. To allow comparison across datasets, we mapped all labels to the common set used in Table 3.4, using the closest label in terms of semantics and structure, *i.e.* Vegetation is mapped to Building and Sidewalk and Grass is mapped to Ground. For *CamVid*, we use 16E5 and 06R0 for training and 05VD for testing, as in (Brostow et al., 2008).

	Gnd	Veh	Ped	Sky	Bld	Avg	Dyn
<i>DUS</i>	79.8	11.3	0.0	26.3	58.1	35.1	5.6
<i>KITTI</i>	67.9	23.7	0.0	42.1	76.3	42.0	11.8
<i>CamVid</i>	89.4	9.1	0.0	60.0	68.2	45.4	4.5
<i>Leuven</i>	75.9	18.3	0.0	52.2	80.7	45.4	9.2

Table 3.4: Comparison of the discussed datasets by means of the static estimator. Higher scores indicate that the scene layout is more static and that training and evaluation parts are more similar to each other. In turn, dataset with lower scores are more complex to solve. The listing is sorted by Avg [IOU](#) performance.

Part I

PIXEL LEVEL

SCENE LABELING AS CLASSIFICATION OF LOCAL IMAGE PATCHES

CONTENTS

4.1	Limitations of the Stixel Model	52
4.2	Local Patch Classification	53
4.2.1	Feature Types	54
4.2.2	Decision Forests for Patch Classification	59
4.3	Extending the Stixel World	62
4.3.1	Updated Graphical Model	63
4.4	Experiments	65
4.4.1	RDF Parameters	65
4.4.2	Feature Type Comparison	66
4.4.3	Runtime	68
4.4.4	Stixel Extension	69
4.4.5	Qualitative Results	74
4.4.6	Comparison to Reference Methods	76
4.5	Discussion	79

Working at the level of pixels has the advantage that there is no preceding processing step involved between image acquisition and the algorithm. Consequently, the accuracy of the results is purely dependent on the algorithms performance and erroneous behavior cannot be attributed to an intermediate step such as superpixel generation. In this chapter, we follow the line of research that approaches scene labeling as classification of local image patches, as discussed in Section 2.4.1. Such a patch is typically rather small and centered around each pixel in the image to capture its local neighborhood. Based on this localized information, a classifier is trained to decide which class label is most likely present behind each pixel.

What sets the work presented here apart from previous work in the literature is the goal of utilizing the inferred scene labeling result to extend the Stixel model from Section 2.2. In this context, the focus of pixel-level inference is not to achieve best classification accuracy for all classes in a full scene labeling setup, but rather to obtain fast and robust results for the subset of classes that are relevant for Stixel generation. Stixels will repeatedly play a central role throughout this dissertation and will later be used as primitive elements the presented methods are built upon. As a result, the quality of the Stixel algorithm will define the lowest level of detail that can be resolved. In this chapter, we therefore discuss some limitations of the current

Stixel model and propose a framework to overcome these limitations, so that Stixels provide a good basis for our designated scene labeling purpose.

To this end, we introduce a pixel-level classification framework that predicts coarse semantic class labels from a set of local image features. We thoroughly study the effect of different feature transformations from the color and disparity image to find the best trade-off between quality and computational complexity. Moreover, we propose how to extend the Stixel formalism to take this new information into account during their estimation process. Parts of this work have previously been published in (Scharwächter and Franke, 2015) and (Cordts et al., 2017b).

4.1 LIMITATIONS OF THE STIXEL MODEL

Earlier in Section 2.2 we introduced the Stixel model and the wide variety of applications it has proven useful for. In this section, we will discuss some of the problems that still exist with this model and that decrease its usability for our designated scene labeling task. The main problems we find can be summarized in three points, which are briefly discussed in the following.

COMPUTED SOLELY FROM DEPTH DATA The most central drawback of the Stixel model as presented by Pfeiffer (2011) is the fact that it only uses depth information as input. In particular if the depth map is generated with a stereo camera, this leads to several problems. Depth resolution in stereo is inherently limited to close and mid range distances, so that object boundaries at far distances cannot be resolved properly. Furthermore, untextured areas cannot be reconstructed, which is then reflected in the Stixel representation by either phantom obstacles that follow the wrong noisy measurements or by not segmenting objects at all.

These two cases then lead to over- or under-segmentation respectively.

LIMITED TO STRUCTURAL CLASSES Without additional information, Stixels can only represent the environment in terms of structural classes: supporting ground plane, perpendicular obstacles, and sky. Any further subdivision is either very complex or not possible at all. Arguably, this representation does not suffice to capture the wide variety of possible street scene configurations. For instance, classes like sidewalk and grass that delimit the free space are missing.

Sky detection only works with textured clouds, where disparity correlation to zero is possible.

TOO STRONG UNDER-SEGMENTATION Given that we would like to use Stixels as primitive elements in our scene labeling concept, they must be fine-grained enough to capture all segment boundaries between classes we are interested in, similar to superpixels. Unfortunately, in its current form, Stixels yield an under-segmentation of

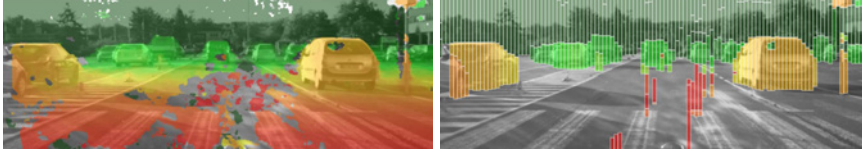


Figure 4.1: Illustration of typical Stixel problems. The left image shows the dense stereo map, the right image shows the resulting Stixels. False matches in the stereo map also generate false positive Stixels. In addition, the background is not separated correctly, *i.e.* tree tops and sky, since color and texture information is not taken into account. Image courtesy of D. Pfeiffer.

the image that is too coarse for us to serve as lowest level for scene labeling. This is also against the principle of least commitment, as discussed earlier in Section 1.2.3. In Figure 4.1, we show typical problems of the existing Stixel algorithm that occur due to the discussed limitations.

A closer look reveals that there is a causal connection between the previous three paragraphs, from the first, over the second, to the last one. As a consequence, this allows us to tackle all problems we described at once by providing additional information to the Stixel algorithm, *e.g.* in the form of appearance-based cues. Therefore, the next section covers how to efficiently extract pixel-level semantic information and how this information can be used in an extended Stixel model that does not suffer from the discussed problems anymore.

4.2 LOCAL PATCH CLASSIFICATION

In the following, our aim is to leverage color, texture, and depth jointly on the level of small local patches to extract the rough geometric and semantic layout of the depicted scene. The relevant contribution in this section is the discussion of various pixel-level feature channels and their combination within a runtime-efficient randomized decision forest framework. Our approach is extremely fast, thanks to modern GPU hardware.

In Section 4.2.1, we discuss different feature transformations from the color and disparity image that are either relevant or specifically designed for visual processing of outdoor images. We supplement this discussion of feature types later in Section 4.4.2, where we provide a thorough evaluation on how much each of the presented transformations actually contributes to the solution of the patch classification problem. Given a meaningful set of features, Section 4.2.2 then covers how to train a classifier that combines these features optimally and yields pixel-level semantic scores that can be integrated into the Stixel estimation process. Note that our approach is developed with the goal of integrating the results into the Stixel framework, but it is in no way limited to this application and could also be used in a differ-



Figure 4.2: Input image and our inferred coarse pixel-level labeling result on the *KITTI* dataset. Our method is able to extract this result in less than 10 ms using a GPU.

ent context, *e.g.* as an attention mechanism for other algorithms. An example result of our inferred coarse semantic class labels is shown in Figure 4.2.

4.2.1 Feature Types

Over the course of a day, the visual appearance of outdoor street scenes is subject to strong variations, depending on weather and illumination conditions. Cast shadows and bright sunlight for instance can have a severe impact on the performance of vision algorithms. It is therefore important to find suitable features that are as invariant as possible w.r.t. to these variations. In the following, we introduce and discuss the different transformations we apply to obtain an informative and complementary set of features. We found that with those features we are able to robustly encode color, texture, and structural information under the constantly changing illumination conditions of outdoor street scenes. We will use the terms *feature type* and *feature channel* frequently throughout this chapter. Typically, one specific feature type yields one feature channel as a result. However, some feature types can result in multiple feature channels, *i.e.* a filter bank with multiple kernels.

4.2.1.1 Color

While color is a strong cue to separate regions such as sky and vegetation, it can vary strongly under different weather conditions or in presence of shadows. In the literature, two notable approaches have been used to gain invariance to illumination changes: rg chromaticity, as discussed in *e.g.* (Gevers et al., 2012) and the illumination invariant image (Upcroft et al., 2014). Both concepts are very simple to execute as they only rely on the RGB values of individual pixels. In the following, both ideas are introduced briefly and their relation is discussed.

RG CHROMATICITY The term chromaticity describes color objectively regardless of its luminance. Hence, it is a very suitable representation for our scenario. The rg chromaticity space is a two-dimensional color space obtained by normalizing the RGB values of each pixel according to

$$r = \frac{R}{R + G + B} \quad g = \frac{G}{R + G + B}. \quad (4.1)$$

In doing so, the absolute intensity information is removed and the blue channel can be dropped, as it can be reconstructed with $b = 1 - r - g$. As a result of this normalization, strong intensity changes such as shadows are attenuated and the colors of foliage and sky are more homogeneous. An example for a normalized image is shown in Figure 4.4 (middle left). In the experiment section (4.4.2), this feature type is called *rgChroma*.

ILLUMINATION INVARIANT IMAGE An interesting transformation has been proposed by Ratnasingam and Collins (2010), who encode chromaticity independent of image intensity and correlated color temperature of the particular daylight spectrum. Under the assumption that the light spectrum is that of a black-body, they define a mapping from RGB to a single-channel illumination invariant image

$$I = \log G - \alpha \log B - (1 - \alpha) \log R, \quad (4.2)$$

where the value α has to be chosen for each camera depending on its sensor characteristics. Upcroft et al. (2014) apply this method for scene labeling and demonstrate that it is more robust compared to directly using RGB values. We follow this approach and call this feature type *illuInv* later. An image visualizing the result of this transformation is shown in Figure 4.4 (middle right).

DISCUSSION The key aspect in both discussed approaches is to remove absolute image intensity and just keep relative color information. Given the quotient rule for logarithms, *i.e.* $\log \frac{a}{b} = \log a - \log b$, it directly becomes apparent that the illumination invariant image used by Upcroft et al. (2014) is in fact equivalent to a ratio of the color channel responses in the log domain. One central difference however is the resulting representation. While the *rgChroma* mapping reduces the three-channel RGB image to two feature channels, the illumination invariant image encodes all information within a single channel only. In Section 4.4.2 it will become apparent that this reduction seems to be too strong, at least in the context of the patch-based classification scheme presented in this chapter. Another interesting alternative to encode color is the opponent color space, which is constructed not by ratios but by differences of R, G, and B (Krauskopf et al., 1982).



Figure 4.3: Filter kernels used for texture encoding. The grayscale image is filtered with each kernel, yielding 8 feature channels.

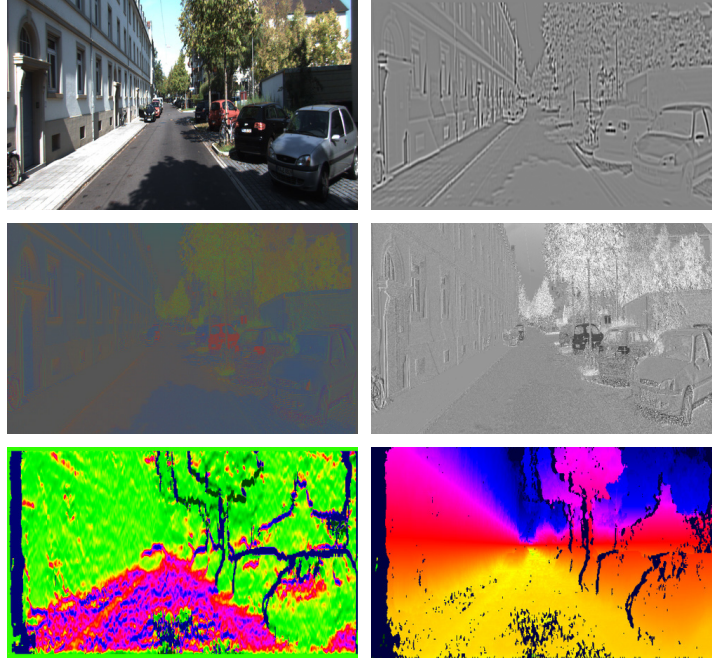


Figure 4.4: Input image (top left) and corresponding feature channels we extract from the color image and the corresponding disparity map. We show one response of the *texture* filter bank (top right), the *rgChroma* image (middle left), *illuInv* image (middle right), and the two depth-based channels *dispGrad* (bottom left) and *hGround* (bottom right) to encode the 3D structure of the scene.

4.2.1.2 Texture

In contrast to color information that has rather low frequency in the image domain, texture encodes local contrast and hence high frequency components. We extract texture information by applying a set of eight filter kernels to the grayscale version of the image. The chosen filters are a subset of the larger filter bank presented by [Leung and Malik \(2001\)](#) and are visualized in Figure 4.3. As apparent from this figure, the chosen filters encode oriented edges as well as blobs and have a support of 13×13 pixels. The filter bank response is called *texture* in all experiments and results in eight feature channels.

4.2.1.3 Depth

In contrast to the raw color information, depth is comparably robust against many environmental conditions as long as sufficient texture

for stereo matching is available. This makes it a rich source of information and complementary to the previously introduced color and texture channels. As it encodes the 3D structure of the scene, it dramatically helps to resolve ambiguities and to avoid a physically implausible labeling. Given the disparity data, we follow the ideas presented by [Brostow et al. \(2008\)](#) and [Zhang et al. \(2010\)](#) and compute two transformations: height above the ground plane and vertical disparity gradient. In the following, we introduce both channels in more detail.

HEIGHT ABOVE THE GROUND PLANE To encode the 3D vertical ordering of objects as well as the fact that objects are physically placed on top of a supporting surface, we transform each pixel into the height above the ground plane, h_{Ground} , using the intrinsic and extrinsic calibration of the camera. More precisely, we use the height component Y_w of the world coordinate point $(X_w, Y_w, Z_w, 1)^\top$, given as

$$\begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & C_x \\ 0 & \cos \phi_p & -\sin \phi_p & C_y \\ 0 & \sin \phi_p & \cos \phi_p & C_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \quad (4.3)$$

where ϕ_p is the pitch angle of the camera, $(C_x, C_y, C_z)^\top$ is the position where the camera is mounted relative to world coordinates and $(X_c, Y_c, Z_c, 1)^\top$ is the 3D point in homogeneous camera-centric coordinates. Coordinates with subscript c correspond to the camera coordinate system, while subscript w denotes world coordinates. The projection from image plane to camera-centric 3D point is given as

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{Z_c}{f_x} & 0 & 0 \\ 0 & -\frac{Z_c}{f_y} & 0 \\ 0 & 0 & Z_c \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -u_0 \\ 0 & 1 & -v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad (4.4)$$

The component Z_c is the 3D distance to the camera and inverse proportional to the measured disparity d , with $Z_c = \frac{f_x b}{d}$. The remaining variables are intrinsic camera parameters and have to be found a-priori by calibrating the stereo camera: $f_x = \frac{F}{s_x}$ is the focal length normalized by pixel size, b is the baseline of the stereo camera rig and (u_0, v_0) is the image focal point. Camera calibration is a challenging task for itself and is not discussed further in this dissertation. A good reference for this topic is ([Burger, 2016](#)).

The camera model is visualized in Figure 4.5. We use this model because the prominent camera motion during normal driving is pitching, either induced by bumps on the road or breaking and acceleration maneuvers. This pitching motion can strongly influence the

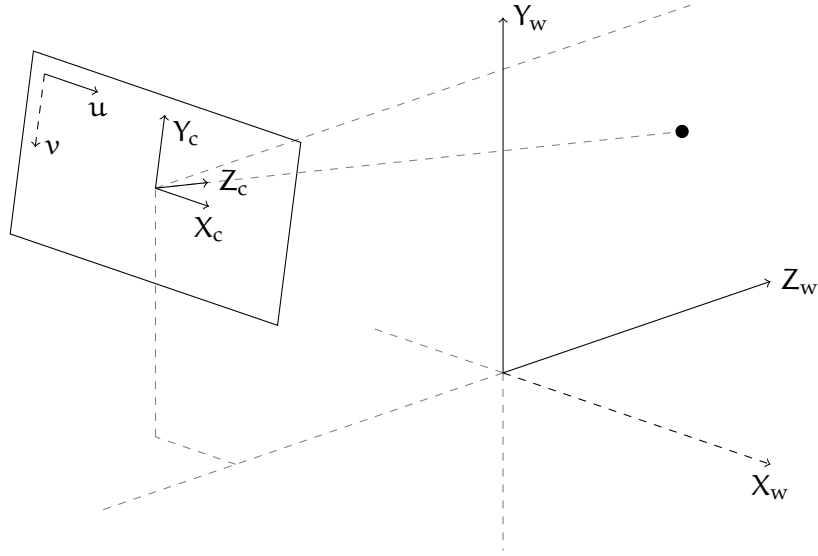


Figure 4.5: Visualization of our camera model. The world coordinate system is located on the ground surface, below the center of the vehicle’s front axle, with Z_w pointing in the direction of driving and Y_w pointing upwards. The camera is mounted at a fixed position relative to this point and is oriented front-facing, without roll and yaw relative to world coordinates. Vehicle pitch ϕ_p is modeled explicitly as rotation around the X_w axis to compensate for changes in the orientation of the camera relative to the ground plane that are caused by acceleration and deceleration.

height estimation of measurements, in particular at far distances. Therefore, the pitch angle ϕ_p of the camera is estimated on a per-frame basis.

VERTICAL DISPARITY GRADIENT Under the assumption of a flat ground surface, the vertical disparity gradient remains constant on the ground surface and only depends on the pitch angle of the camera (Labayrade et al., 2002). This fact makes the gradient a strong feature that is mostly valuable for ground plane detection. Note that the gradient signal is less sensitive to small deviations from a flat ground model compared to *hGround*. To obtain a robust estimate of the vertical gradient, we adopt a 31×3 window and only take valid measurements into account. We call this type *dispGrad* in the following. Note that the gradient channel encodes information similar to the surface normal feature from Brostow et al. (2008), but is much faster to compute. Figure 4.4 (bottom) shows an example of the two channels extracted from the disparity image.

4.2.1.4 Other

PIXEL LOCATION To encode spatial layout for pixels without valid depth measurements, we additionally employ the feature type *pixel-Loc*, containing the pixels' vertical distance to the horizon line.

INTENSITY As our final feature, we use the raw single-channel *intensity* image to encode that some classes do have a typical repeating intensity level during daylight.

In summary, we could of course use more elaborate feature transformations, but the presented set of features offers a good trade-off between expressiveness and computational costs, from an application point-of-view.

4.2.2 Decision Forests for Patch Classification

A seamless integration into the probabilistic Stixel framework requires soft per-class probabilities for each pixel rather than a hard decision that just returns the most probable label. To infer semantic labels from the feature maps presented above, we follow [Fröhlich et al. \(2012\)](#); [Müller and Behnke \(2014\)](#); [Shotton et al. \(2008\)](#) and use a Randomized Decision Forest (RDF) classifier for pixel-level scene labeling. Note that the choice of classifier is not essential for the problem at hand, but in contrast to SVMs or CNNs, RDFs are favored here as they allow for fast inference time and directly provide empirical class posterior distributions as output that can be used within the Stixel algorithm without the need to transform the classifier scores to probabilities. Furthermore, they can be executed efficiently using modern GPU hardware. In the following, we outline the general idea behind randomized decision forests and provide some mathematical background on how they are trained.

4.2.2.1 General principle of RDFs

Randomized decision forests can be seen as an extension of classical decision tree classifiers. A decision tree is a classifier that uses a tree structure as predictive model. This tree structure is typically binary, so that for a given test sample a yes/no question is asked at each node. Depending on the answer, the sample is then passed to the left or right child node respectively. This procedure is continued until the sample reaches a leaf node, in which the class decision, or more frequently, a posterior probability distribution over the classes is stored. An example of a decision tree classifier is given in Figure 4.6. This method of classification is straightforward and effective, but it has some practical drawbacks such as the habit of overfitting to the training data. To address this problem, [Breiman \(2001\)](#) introduced "Random Forests" that combine his idea of bootstrap aggregation, or *bag-*

The term "Random Forest" is an official trademark of Leo Breiman.

ging, with the random subspace selection of Ho (1998). Bagging is a machine learning meta-algorithm for ensemble classifiers. The idea is quite simple: instead of training one classifier on all training samples S , an ensemble of L classifiers is trained on L different subsets of the training data, where each subset is generated by sampling uniformly and with replacement from S . Finally, the classifiers are combined by averaging their results. In this way, the variance of individual classifier outputs is reduced and overfitting is mitigated. The random subspace selection is similar in spirit to bagging, but instead of randomly choosing a subset of samples, a subspace within the typically high-dimensional feature or attribute space of a sample is chosen. In fact, random subspace selection is also called *attribute bagging*. A useful side-effect of attribute bagging is further that it makes the method largely independent of the feature dimensionality, which can be very important in practice. Both strategies are complementary and used together in randomized decision forests to maximally reduce variance and overfitting.

In this chapter, we are interested in a classifier response for each pixel in the image. To this end, we train and evaluate the RDF based on a local patch around a pixel, so that the input feature vector contains the pixel-level responses of all feature channels from Section 4.2.1 within this local patch. Figure 4.6 shows an example of a multi-channel image patch, centered around a pixel of interest p . It is easy to see that the resulting feature vector can quickly become very high-dimensional. For instance, enabling all feature types discussed in Section 4.2.1 yields 15 channels. Together with a patch size of only 11×11 pixels, as in Figure 4.6, this results in a theoretical 1815-dimensional feature space the pixels are embedded in. It is easy to see now that the attribute bagging of RDFs is highly relevant in practice, as only a very small fraction of these dimensions is ever inspected during test time. In the toy example from Figure 4.6, only 4 out of 1815 dimensions are inspected. This example directly brings us to the time complexity of RDFs during testing. In case we have a decision forest with T trees and limit each tree to a maximum depth of D , complexity to classify a single sample is $O(TD)$ in Landau notation, *i.e.* it is independent of the feature dimension K . Other classifiers such as SVMs are linear or even quadratic in K , making them less useful for our task.

4.2.2.2 Training procedure

We will now take a look at how an RDF model is built. In general, RDFs can be trained in supervised and unsupervised mode, depending on how decisions are found during training. However, in the scope of this dissertation, we will always use RDFs in a supervised setting, where the target label is available during training. As the training procedure is identical for all trees in the forest, it is suffi-

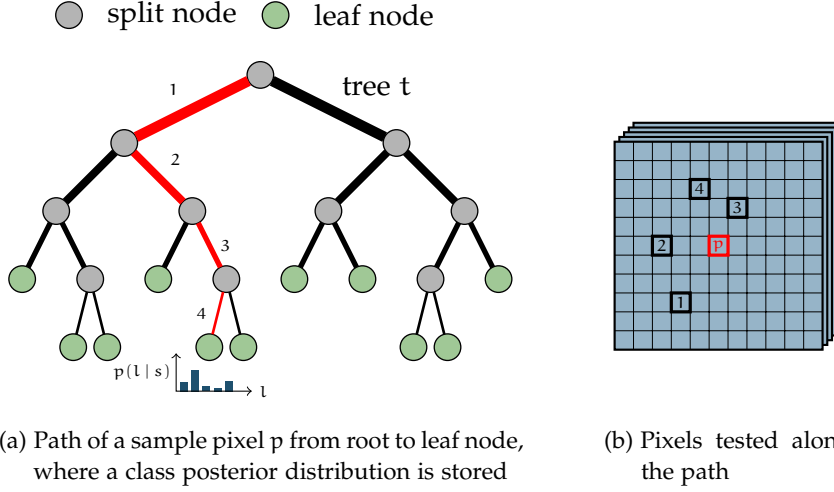


Figure 4.6: Example of a decision tree applied to pixel-level classification. For a sample s of pixel p , each node compares the value of a pixel in a specific feature channel against a learned threshold. Depending on the outcome, traversal continues at the left or right child node, until a leaf node is reached. The leaf node then contains a posterior distribution over the class labels $p(l | s)$. Pixels 1, 2, 3, and 4 inspected along the path are within a patch around the pixel of interest p .

cient to describe how a single tree is grown. The only difference between the trees is the set of training samples used to grow them, as discussed before. Now let $S = \{s_1, s_2, \dots, s_n\}$ be the set of training samples selected randomly for one specific tree, where one sample $s_i = (l_i, \mathbf{x}_i = (x_1, x_2, \dots, x_K))$ is the pair of true class label l_i and K -dimensional feature descriptor \mathbf{x}_i . Without loss of generality, the multi-channel image patch can always be described by means of a $1 \times K$ vector.

Training of a tree is performed recursively, starting at the root node. At each node, the training samples are divided by a randomly chosen, axis-aligned split function. Subsequently, the split function is scored to see how much it improves label purity in the subsets. This process is repeated A times and the split function with highest score wins. The resulting two subsets are then passed to the left and right child node and the process continues. Tree growing ends if all samples in the set have the same label or maximum depth D is reached. The tree growing algorithm in pseudo-code is given in Algorithm 4.1. To score the different split functions, we use information gain $I(S, \mathcal{T})$, defined as

$$I(S, \mathcal{T}) = H(S) - \sum_{p \in \{l, r\}} \frac{n_p}{n} H(S^p) \quad (4.5)$$

where $H(S)$ is the class label entropy of the training set before splitting and $\frac{n_l}{n}$ the fraction of samples in the left and right subset after splitting. The entropy is given as

$$H(S) = - \sum_{l=1}^L \frac{n_l}{n} \log \frac{n_l}{n} \quad (4.6)$$

with $\frac{n_l}{n}$ being the fraction of samples with label l in the set S . For more background on the training procedure, we refer to [Moosmann et al. \(2008\)](#) and [Geurts et al. \(2006\)](#).

Algorithm 4.1 Decision tree training.

Input: Training set S

Output: Grown tree t

Parameters: number of split trials A

```

procedure TRAINNODE( $S$ )
  if STOPSPLITTING( $S$ ) then    | if one stopping criteria fulfilled
    return CREATELEAFNODE( $S$ )
  end if
  for all  $a \leq A$  do
    choose split feature dimension  $k_a$  randomly
    choose split threshold  $\theta_a(k_a)$  randomly
    define split test  $\mathcal{T}_a : x_{k_a} < \theta_a$ 
    split training set according to  $\mathcal{T}_a$ :   $S_a^l = \{s \in S \mid x_{k_a} < \theta_a\}$ 
                                            $S_a^r = \{s \in S \mid x_{k_a} \geq \theta_a\}$ 
    compute score  $I_a(S, \mathcal{T}_a)$     | information gain
  end for
  choose split  $(k_{\text{best}}, \theta_{\text{best}}, S_{\text{best}}^l, S_{\text{best}}^r)$  with highest score
   $\mathcal{N}^l \leftarrow \text{TRAINNODE}(S_{\text{best}}^l)$     | grow left subtree recursively
   $\mathcal{N}^r \leftarrow \text{TRAINNODE}(S_{\text{best}}^r)$     | grow right subtree recursively
  return CREATEDECISIONNODE( $k_{\text{best}}, \theta_{\text{best}}, \mathcal{N}^l, \mathcal{N}^r$ )
end procedure
 $\mathcal{R} \leftarrow \text{TRAINNODE}(S)$     | the root node of tree  $t$ 

```

4.3 EXTENDING THE STIXEL WORLD

The local patch classification scheme from Section 4.2 yields a posterior probability image for each semantic class label, *c.f.* Figure 4.7. In this section, we will extend the Stixel formalism introduced in Section 2.2 to take these class-specific probability maps into account, in addition to stereo depth maps. To this end, we further split up the three *structural classes*, “support” (\mathcal{S}), “vertical” (\mathcal{V}), and “sky” (\mathcal{Y}) from Section 2.2 into *semantic classes* that are mapped onto the sets \mathcal{S} , \mathcal{V} , or \mathcal{Y} . Semantic classes such as road or grass, for example, are in the support set \mathcal{S} , whereas building, tree or vehicle are vertical, *i.e.*

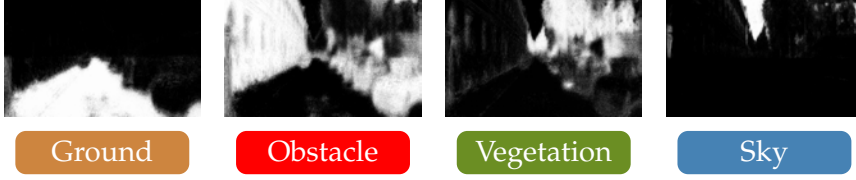


Figure 4.7: Probability maps for semantic class labels in a four class setup. The result is computed for the same input image as in Figure 4.4. Note that our approach is able to recover from erroneous or missing depth measurements on the ground plane or in the sky, *c.f.* Figure 4.4 (bottom).

in \mathcal{V} . The actual set of semantic classes used will be introduced and discussed later in Section 4.4.4.

4.3.1 Updated Graphical Model

Figure 4.8 shows an extension of the graphical model shown in Figure 2.4 that was used to introduce the original Stixel framework (*c.f.* Section 2.2). Compared to Figure 2.4, we introduce a new factor to the likelihood term that is connected to the semantic label input data L , in addition to the depth map input data D . The detailed modifications of the inference algorithm in terms of likelihood and prior term and their implications are discussed in the following two paragraphs.

DATA LIKELIHOOD As shown in Figure 4.8 (right), the measurements \mathbf{M} , now consist of a disparity map \mathbf{D} , and semantic label score maps \mathbf{L} , (one map for each label). To use this additional input, the likelihood term in Equation (2.14) is extended with an additive factor $\Phi_L(\cdot)$, yielding

$$\Phi(\mathbf{s}_i, \mathbf{m}_i) = \sum_{i=1}^h \sum_{v=v_i^b}^{v_i^t} \Phi_D(\mathbf{s}_i, d_v, v) + \Phi_L(\mathbf{s}_i, \mathbf{l}_v) . \quad (4.7)$$

This new factor is defined as

$$\Phi_L(\mathbf{s}_i, \mathbf{l}_v) = -\delta_L(c_i) \log(l_v(c_i)) , \quad (4.8)$$

and consists of a class-specific weight factor $\delta_L(c_i)$ and the labeling scores $l_v(c_i)$, converted to costs in the log-domain. Note that the scores are normalized, with $\sum_{c_i} l_v(c_i) = 1$ for all considered classes c_i at all pixels v .

A particularly important aspect of this additive extension is that both factors can complement each other, *i.e.* both can instantiate new Stixels independently, if the other factor is agnostic about it (constant value). Consider the following example: The depth-based factors of *e.g.* Building and Tree are identical, as they both belong to the struc-

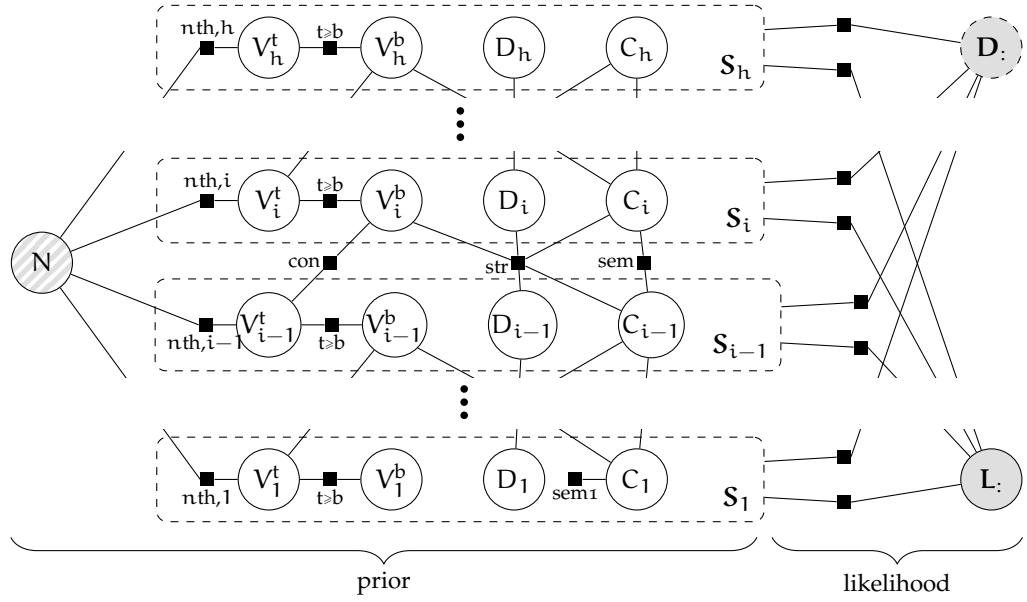


Figure 4.8: Extension of the factor graph in Figure 2.4, where semantic label scores are added as additional input, *c.f.* right circle L (likelihood), but the prior factors remain unchanged. Courtesy of Marius Cordts.

tural class “vertical”. The same applies to Grass and Road, which belong to the structural class “support”. As a result, new Stixels can be introduced to follow a label change, purely based on the semantic factor. In the same way, the pixel-level classifier might deliver a generic Vegetation label that is agnostic about the two structural classes “support” and “vertical”. In this case, we can instantiate two Stixels that split this label into Tree and Grass, purely based on the depth factor.

PRIOR The factors that contribute to the prior term are in fact the same as in the original Stixel formulation, because it already contains a factor $\Psi_{\text{sem}}(\cdot)$ that models prior knowledge about label transitions, *c.f.* Equation (2.13). The only difference is that the original Stixel formulation considered three structural classes and the transitions between them, but our extended model solves for an arbitrary number of semantic classes. Consequently, the two-dimensional transition matrix $\gamma_{c_i, c_{i-1}}$ from Equation (2.13) grows quadratically with the number of classes involved. As all possible transitions must be evaluated during inference, the execution time of the Stixel extension proposed here increases as well. To avoid this complexity increase, Lukas Schneider proposes an alternative way of considering semantic class labels in the Stixel algorithm (Cordts et al., 2017b).

4.4 EXPERIMENTS

The evaluation in this chapter covers three major aspects. We first provide an in-depth performance analysis of the proposed pixel-level classification scheme. In particular, we sweep the central **RDF** classifier parameters, study which of the discussed feature transformations are most informative to support the task of local patch-based classification, and analyze the runtime of our approach. The second aspect is the integration of our inferred pixel-level label scores into the Stixel model, where we will find that our extension improves the representation. The extended algorithm then also outperforms the unary pixel-level labeling result due to the regularizing effect of the model. Finally, we compare the results to reference methods presented in the literature.

4.4.1 RDF Parameters

Before we start with quantitative results, we will briefly outline the experimental setup and practical training details. For the results shown here and in Section 4.4.2, the pixel-level random forest is trained and evaluated on a subset of image patches that have been extracted from the original dataset images with a horizontal and vertical stride of 16 pixels. This reduction is done to keep the number of samples manageable in practice. During the tree growing process, each tree starts with a 5% subset of the training samples, randomly selected, to prevent correlation between different trees. This subset is then recursively subdivided by new split nodes until the classification problem on this subset is solved completely. To find a good split function for a node, we choose the best out of 1000 independent split trials, as outlined in Algorithm 4.1. As the trees are trained to solve the classification problem perfectly on their small subset, they are now overfitted to this subset and have poor generalization performance. To improve generalization and to obtain balanced and statistically robust class posteriors, we then update the class distribution at each leaf node based on all training samples, *i.e.* we let all samples traverse the trees and compute the class label statistics based on all samples that arrive at each leaf node. Subsequently, we further prune leaf nodes if the class distribution is generated from less than 20 samples. During test time, the label with the highest classifier score is assigned to each pixel. Also, if not stated otherwise, all experiments have been conducted 5 times due to randomness in the training procedure. The shown numbers thus correspond to the average performance over these runs.

We analyze the influence of the two central **RDF** hyperparameters, *i.e.* the number of trees and the maximum tree depth. These two parameters also have most impact on the inference time. In addition, we also vary the patch size to test whether more image content around

This is equivalent to a labeling based on hamming loss.

the pixel of interest helps to improve classification performance. Figure 4.9 shows the average *IOU* score over different parameter choices. For this experiment we enable all discussed feature types. As base configuration, we use 15 trees, a patch size of 33×33 px and a maximum tree depth of 20. While the parameter of interest is varied, the other two parameters are kept fixed to these values. The plots in Figure 4.9 reveal that for all three parameters, the performance quickly reaches a plateau level and then only improves very slowly. In the case of patch size, performance even decreases for larger windows. We see that a maximum depth of 10 is sufficient if all feature types are used and 15 trees offer a reasonable trade-off between accuracy and runtime. In case of patch size, the performance reaches its maximum with 88×88 patches on the *DUS* dataset and 33×33 patches on the *KITTI* dataset. Our intuition is that the larger patches are required to compensate the lack of color information in the *DUS* dataset, as a small gray value patch is less informative than a patch of color data. Based on the observation that the curves are relatively flat except for very extreme *RDF* parameter choices, we conclude that the features seems to be more important than the *RDF* classifier itself. The classifiers purpose is merely to combine the features and map them to probabilistic scores.

4.4.2 Feature Type Comparison

Now that we determined that the features are a crucial part of our approach, we continue with the next experiment to obtain a clearer picture about which type of feature contributes how much to the overall accuracy. For this experiment, we setup the base *RDF* configuration from before, *i.e.* 15 trees, a 33×33 patch and a maximum tree depth of 20. The high tree depth is used to account for the fact that some feature type *A* might need more split tests than some type *B* but would still give comparable performance in the end. Furthermore, despite the fact that we found a better patch size for the *DUS* dataset, we decide to set it equal for both datasets here. This is done to obtain a consistent ranking of feature types across the two datasets and with that a more reliable result.

We start by using all feature types together to estimate the accuracy of the full model. The resulting pixel-level confusion matrices for the *DUS* and *KITTI* datasets are shown in Table 4.1 and Table 4.2 respectively. The confusion matrices of the full model already reveal that some classes cannot be solved sufficiently well. More specifically, we observe strong confusion between the Vehicle and Pedestrian label and additionally the Ground and Sidewalk label on the *KITTI* dataset.

To assess the performance of each individual feature type, we now follow a leaving-one-out strategy, where a strong feature type yields a large drop in performance when left out. Additionally, we report

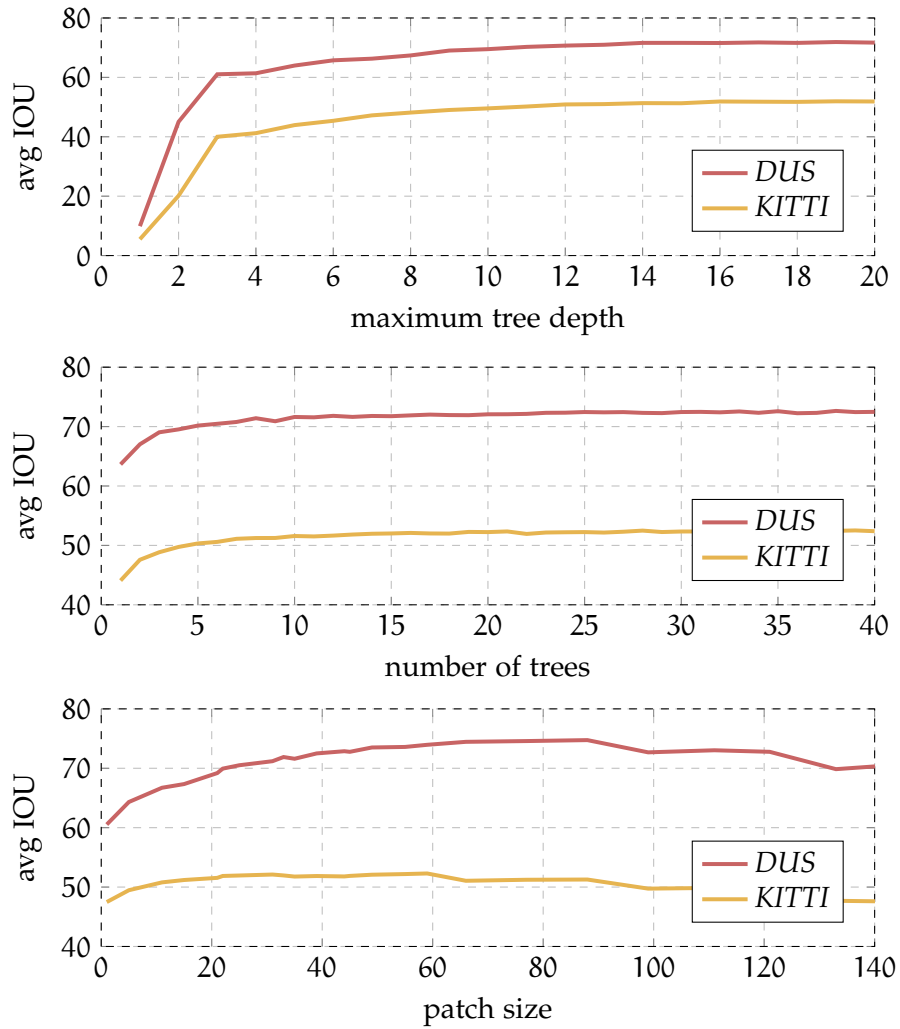


Figure 4.9: Evaluation of the three central [RDF](#) parameters: tree depth, number of trees, and patch size accessible to the [RDF](#). We conducted this experiment on the *DUS* and *KITTI* dataset and show the average over five independent runs to reduce the effect of randomness in the method.

		predicted				
		Gnd	Veh	Ped	Sky	Bld
actual	Gnd	96.3	2.4	1.3	0.0	0.1
	Veh	0.5	81.3	13.3	0.0	4.9
	Ped	1.7	35.3	59.9	0.1	3.1
	Sky	0.0	0.0	0.0	99.4	0.6
	Bld	0.0	3.5	3.6	3.2	89.7
	IOU	95.9	62.8	38.1	75.4	86.2

Table 4.1: Pixel-level confusion matrix on the *DUS* dataset using all feature channels. The corresponding class-specific IOU scores are provided in the last row.

results when using only a single feature type in isolation. The results of this experiment are shown in Table 4.3 and Table 4.4. The IOU measure is used for all numbers in these two tables. The results clearly indicate that individual feature types alone deliver poor performance, but the combination in a joint model yields good results. Furthermore, the depth-based features increase performance significantly while 3D height above the ground plane, *hGround*, seems to be the most important feature. Regarding color, we see that the *illuInv* channel is less informative than the two *rgChroma* channels in our setup. We attribute this to the reduction of color information from two channels to only one channel, as discussed earlier in Section 4.2.1.1. Overall, the ranking of feature types is largely consistent across both datasets. We see that *intensity* has more weight on the *DUS* dataset, as it must compensate for the lack of color. Surprisingly, *texture* contributes less in our setup. In fact, removing *texture* completely even improves performance slightly. We attribute this to the low performance of the *texture* feature type in combination with its large number of channels. As more than half of all feature channels result from the *texture* filter bank, namely 8 out of 15, chances to select one of these channels during *RDF* training are much higher. Consequently, potentially more informative feature types are not chosen. A solution to this problem would be to randomly select a candidate feature *type* first, before the actual candidate feature *channel* is chosen or to directly choose *texture* channels with smaller probability. Alternatively, the *texture* features could simply be left out completely.

4.4.3 Runtime

We implemented our approach on *GPU* using the NVIDIA CUDA toolkit. Figure 4.10 shows the average runtime to classify a *KITTI*

		predicted						
		Gnd	Veh	Ped	Sky	Bld	Sid	Veg
actual	Gnd	80.0	2.0	0.1	0.0	0.0	17.6	0.2
	Veh	0.2	85.2	5.9	0.5	4.5	2.2	1.4
	Ped	0.0	54.2	41.8	0.0	2.0	1.6	0.4
	Sky	0.0	0.0	0.0	93.4	4.0	0.0	2.6
	Bld	0.1	13.3	2.3	4.0	73.5	0.8	5.9
	Sid	35.7	7.3	0.4	0.0	0.0	55.8	0.8
	Veg	0.2	10.5	2.0	0.4	8.8	2.4	75.7
IOU		68.5	53.4	2.5	66.3	67.0	36.1	69.8

Table 4.2: Pixel-level confusion matrix on the *KITTI* dataset using all feature channels. The corresponding class-specific IOU scores are provided in the last row.

image with an NVIDIA GTX 770 GPU. This runtime includes transfer from and to the GPU, computation of all feature channels and RDF classification. Timings are reported for different sub-sampling factors of the image. A factor of 1 means operation at native image resolution, which is about 0.5 mega pixels for *KITTI*. Note that the factor applies to image width *and* height at the same time, so that the number of pixels reduces quadratically with lower values.

It can be seen that the overall runtime is dominated by the RDF classification and only a very small part is spent on feature computation. At the same time, sub-sampling the image can drastically reduce the time spent on classification, while transfer and feature extraction almost remain constant. Overall, we find that a sub-sampling factor of 3 gives a reasonable trade-off between runtime and labeling resolution. For this experiment, we train 15 trees to a maximum depth of 10 to optimize runtime. Again, we enable all feature channels to remain consistent to the previous experiment. However, by disabling the *texture* feature channels, we can further reduce runtime roughly by half without losing accuracy, as discussed earlier.

4.4.4 Stixel Extension

In the following, we will evaluate how our proposed Stixel extension improves the original variant published by Pfeiffer (2011). Before we do so however, let us briefly summarize our findings so far. The previous experiments clearly show the potential but also the limits of our proposed local patch classification scheme. It is very fast to compute and yields a good coarse semantic labeling of the scene. However, some class labels, in particular Vehicle and Pedestrian, suffer from

channel	Gnd	Veh	Ped	Sky	Bld	Avg
all channels						
	95.9	62.8	38.1	75.4	86.2	71.7
this channel left out						
<i>texture</i>	96.1	63.6	39.1	75.2	85.7	72.0
<i>pixelLoc</i>	95.5	62.4	40.3	74.3	85.5	71.6
<i>hGround</i>	95.2	59.1	38.5	77.8	84.9	71.1
<i>dispGrad</i>	95.1	60.6	34.6	74.8	83.7	69.8
<i>intensity</i>	94.5	58.0	31.8	64.5	84.6	66.7
depth channels left out						
	90.6	51.6	33.3	76.8	79.1	66.3
depth channels alone						
	93.8	56.6	31.1	38.4	75.7	59.1
this channel alone						
<i>hGround</i>	91.5	49.0	22.7	36.8	72.1	54.4
<i>intensity</i>	78.6	41.5	24.8	73.1	43.0	52.2
<i>dispGrad</i>	93.3	40.2	23.1	26.2	62.3	49.0
<i>texture</i>	67.3	21.4	19.3	77.2	45.8	46.2
<i>pixelLoc</i>	70.4	15.5	12.5	17.8	47.1	32.7

Table 4.3: Pixel-level classification results on the *DUS* dataset for different feature channel combinations. Lines are sorted block-wise by average *IOU* score in descending order (right column).

channel	Gnd	Veh	Ped	Sky	Bld	Sid	Veg	<i>Avg</i>
all channels								
	68.5	53.4	2.5	66.3	67.0	36.1	69.8	51.9
this channel left out								
<i>texture</i>	68.3	53.6	2.4	67.0	66.8	36.6	69.9	52.1
<i>illuInv</i>	68.4	53.5	2.5	65.6	67.1	36.1	70.1	51.9
<i>intensity</i>	68.9	53.1	2.6	68.0	65.6	36.1	66.1	51.5
<i>rgChroma</i>	68.6	52.4	2.5	64.6	67.1	36.5	68.3	51.4
<i>dispGrad</i>	69.1	51.5	2.0	66.0	65.9	33.5	70.1	51.2
<i>pixelLoc</i>	67.1	52.4	2.8	64.1	66.1	35.2	69.9	51.1
<i>hGround</i>	55.7	48.6	2.4	45.4	60.8	33.0	70.4	45.2
color channels left out								
	67.9	50.2	1.4	61.9	61.3	33.4	50.4	46.7
depth channels left out								
	47.9	40.3	1.8	44.4	59.7	27.5	71.1	41.8
depth channels alone								
	67.1	49.0	1.4	47.3	47.3	34.0	15.4	37.4
this channel alone								
<i>hGround</i>	66.4	45.9	1.1	43.0	46.4	32.2	11.4	35.2
<i>rgChroma</i>	39.9	31.1	0.7	30.6	24.4	14.0	68.7	29.9
<i>illuInv</i>	42.7	24.3	0.8	29.7	16.8	10.3	65.0	27.1
<i>dispGrad</i>	55.2	27.3	1.0	11.4	38.3	30.6	2.7	23.8
<i>intensity</i>	30.7	21.3	0.4	23.3	21.8	18.3	37.6	21.9
<i>texture</i>	24.9	8.3	0.3	17.3	27.4	14.3	34.6	18.2
<i>pixelLoc</i>	24.8	7.7	0.4	12.9	26.2	16.1	0.0	12.6

Table 4.4: Pixel-level classification results on the *KITTI* dataset for different feature channel combinations. Lines are sorted block-wise by average *IOU* score in descending order (right column).

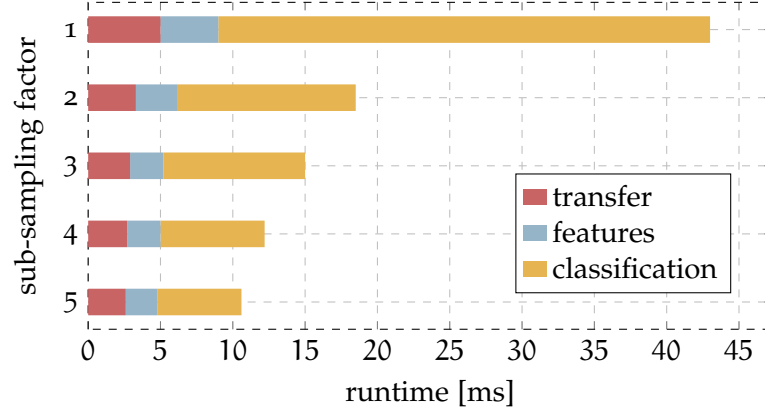


Figure 4.10: Average runtime of the pixel-level classifier to densely label a *KITTI* image. Timings are broken down to three components: data transfer from and to GPU, feature extraction and classification. A sub-sampling factor of 1 means native image resolution. Disabling the *texture* channels further cuts runtime roughly by half.

The more class labels, the harder it becomes to achieve high scores for all classes.

high confusion. Arguably, this is a problem, given that those two labels are most relevant for autonomous vehicles. As introduced earlier in Section 1.2.3, one of the ideas put forward in this dissertation is to incrementally build a system from basic building blocks, where each component should provide additional information as much as it is capable of, not less and not more. We would call a scene labeling method to be accurate, if it provides an average *IOU* of about 90.0 for *DUS* or about 75.0 for *KITTI*. We therefore now combine some of the class labels with highest confusion to obtain a set of coarse labels, so that the average performance is in the desired range of accuracy. Only these labels are then integrated into the Stixel framework, so that the extended model improves over previous Stixels, but does not suffer from the limits of the patch-level classifier. Consequently, we postpone the full labeling problem to the next chapter on region-level methods. The coarse set of labels is obtained by combining the Building, Vehicle, and Pedestrian labels into a generic Obstacle label. For the *KITTI* dataset, we further assign the Sidewalk label to Ground and train the classifier to only infer the reduced label set. The resulting confusion matrices are given in Table 4.5.

We test the performance of our Stixel extension by showing the labeling result of Stixels before and after our modification. Additionally, we compare the labeling performance of the extended model with the direct pixel-level labels. The results are summarized in Table 4.6. We perform this experiment with two resolution settings. The upper part of the table shows results reported at native image resolution. We compute our extended Stixels without any reduction in scale, so that we obtain Stixels with a width of one pixel. This is to maintain consistency with Table 4.5 and to demonstrate the regular-

					predicted				
					Gnd Sky Obs Veg				
actual	Gnd	97.1	0.0	2.9	Gnd	95.9	0.0	3.4	0.7
	Sky	0.0	99.5	0.5	Sky	0.0	93.5	3.6	2.8
	Obs	1.3	1.9	96.8	Obs	2.2	4.0	85.8	8.0
					Veg	1.9	0.4	14.6	83.1
	IOU	96.0	74.0	93.5	IOU	89.7	61.4	78.2	71.8

(a) *DUS* dataset
(b) *KITTI* dataset

Table 4.5: Pixel classifier confusion matrix using all feature channels. The corresponding class-specific IOU scores are provided in the last row. The results correspond to Table 4.1 and Table 4.2, except that the Building, Vehicle, and Pedestrian labels have been mapped to a generic Obstacle label and the Sidewalk label is assigned to Ground for use in Stixels.

ization effect of Stixels without any loss induced by down-sampling. However, as working at native image resolution is impractical regarding runtime, we then switch to a typical working resolution in the lower part and compare the original Stixel result with our extension. Images are reduced horizontally and vertically by a factor of 3 for the pixel classifier and then further reduced horizontally by a factor of 2 for Stixels, so that we obtain Stixels with a width of 6 pixels.

The results in Table 4.6 clearly indicate that that our pixel-level label scores improve Stixels, *i.e.* our extended model consistently outperforms the original Stixel model on all classes. At the same time, the extended model also outperforms the direct unary pixel-level labels from our classifier. This shows that the Stixel algorithm is able to make use of the classifier’s uncertainty and to regularize the result. Finally, we can also see that scaling down the image to our practical working resolution only results in a very small performance drop. All these results can consistently be observed on both datasets.

Although the original Stixel algorithm solves explicitly for the Sky label, assuming that sky segments have a disparity close to zero, the performance shown here is poor on both datasets. This is because sky areas in both datasets are very homogeneous and untextured, leading to many false or missing disparity measurements. In these cases, the algorithm tends to generate a single obstacle segment instead and thus misses the sky completely. Furthermore, there is an important aspect to consider when interpreting the *KITTI* results. The reason why original Stixels seemingly perform inferior is that they do not solve for the Vegetation class at all. All vegetation areas are either detected as Obstacle (*e.g.* trees) or Ground (*e.g.* grass), which accounts

	Gnd	Sky	Obs	Avg
pixel classifier ¹	96.0	74.0	93.5	87.8
ext. Stixels ¹	96.8	86.4	95.4	92.9
orig. Stixels ²	95.1	0.1	88.8	61.3
ext. Stixels ²	96.8	84.1	95.2	92.0

¹native image resolution, no scale down

²working resolution, see text

(a) *DUS* dataset

	Gnd	Sky	Obs	Veg	Avg
pixel classifier ¹	89.7	61.4	78.2	71.8	75.3
ext. Stixels ¹	89.2	73.3	80.3	72.7	78.9
orig. Stixels ²	76.5	1.1	58.6	0.0	34.0
ext. Stixels ²	89.1	71.9	79.9	71.5	78.1

¹native image resolution, no scale down

²working resolution, see text

(b) *KITTI* dataset

Table 4.6: Performance evaluation of our Stixel extension. We show the [IOU](#) labeling accuracy of original Stixels, our extension and the direct pixel-level classifier result on the *DUS* and *KITTI* datasets. The pixel classifier results correspond to the numbers in Table [4.5](#).

for the low scores on these two classes, in addition to the Vegetation score of 0.0 and the poor sky performance. However, given that depth alone only allows to infer structural classes, this behavior is in fact correct. Nevertheless, the experiment shows that we could successfully extend the structural classes with semantic classes from our pixel classifier. Qualitative results will be shown in the next section.

As a final remark, we found that it is very important for the Stixel extension to update the leaf node statistics of the decision trees with all training samples, as discussed earlier in Section [4.4.1](#). Only then, the pixel-level probability maps are balanced and smooth enough, allowing the Stixel model to regularize the result. Without this update step, results of the pixel classifier have too much confidence, so that many errors simply overrule the Stixel model constraints and are adopted in the Stixel result.

4.4.5 Qualitative Results

In Figure [4.11](#) and Figure [4.12](#), we show example results of our pixel-level classification approach on the *DUS* and *KITTI* dataset respec-

tively. Together with the input image, the depth image is provided to get a better feeling for the data and to anticipate the behavior of the Stixel algorithm that originally finds segments from depth only. The center images show our raw local classifier prediction without any kind of regularization involved. The color intensity encodes uncertainty in the decision, *i.e.* at boundaries between different classes where stronger confusion is intuitive. We illustrate the performance when inferring the full set of classes (fine) as well as the reduced set of classes (coarse) that is eventually used during our extended Stixel optimization. Finally, the right column shows Stixels in their original form as presented in Pfeiffer (2011), as well as our extension that takes into account our labeling result.

The shown examples illustrate well, where the inferred labels are accurate and also where the limits of the approach are. Overall it can be observed that coarse scene structure is reconstructed quite accurately. We find that depth information mostly helps to separate elevated obstacles from the ground plane, while color and texture are informative to detect sky and vegetation.

Considering the full label set (center top images), the results are promising, but not sufficiently accurate in the sense discussed before. While there certainly is a tendency towards correct separation of vehicles and pedestrians, the result contains many hallucinated false positives for these classes. This happens particularly in the relevant height range from 0 m to 2 m above the ground plane, where vehicles and pedestrians typically occur, *c.f.* Figure 3.6. We also observe that below a certain 3D height value almost no buildings are inferred at all, even if there is actually a building present in the scene. Furthermore, most sidewalks are falsely detected at the boundary to elevated obstacles, *i.e.* when there is a subtle elevation from the 3D ground plane. All these findings indicate how strongly the 3D information influences the decision process, both positively but also negatively. They demonstrate that detecting *all* classes with high accuracy seems to be an ill-posed problem on the local pixel level we operate on in this chapter and justifies the use of models that use more spatial support in the image, as will be discussed later in this dissertation.

In terms of our Stixel extension, the most prominent difference to the original Stixel result is the improved detection of sky. A closer look further reveals better object boundary adherence than before. Detecting sky properly based on depth information alone is difficult and only possible if the sky is textured, *i.e.* clouds are present, so that correct stereo correspondences can be found. In many scenarios however this is not the case, including the two datasets we tested on. For the *KITTI* results, the differences between the original Stixel result and our extension are more pronounced due to the inclusion of the vegetation labels. In particular notice the different shades of green that separate flat vegetation such as grass from upright stand-

ing vegetation such as bushes and trees. Although the pixel classifier delivers only a single vegetation label, the Stixel optimization with its two competing plane models (ground and upright standing) allows to make this differentiation, *c.f.* Section 4.3.

4.4.6 Comparison to Reference Methods

In the following, we compare the results of our extended Stixel model with the reference methods discussed earlier in Section 2.4.3. The Stixel performance numbers are copied from Table 4.6a, where Stixels have been computed at the typical working resolution. The performance numbers for the reference methods are generated by training with the full set of classes and then mapping the result to the coarse set of classes. This is done to achieve consistency between the reference numbers presented here and the numbers presented later in Section 5.3.5.

Table 4.7 and Table 4.8 show the result of the comparison for the *DUS* and *KITTI* dataset. From the numbers it directly becomes apparent that our extended Stixel model is competitive with the reference methods. In particular, it outperforms all other methods on the Ground label, which can be attributed to the use of depth information and the regularization effect of the Stixel model. For the remaining classes, Stixels perform below average. In particular the labeling performance of Sky is considerably lower compared to the reference methods, which can partially be attributed to low performance of our pixel-level classifier, in particular on the *KITTI* dataset. One reason for this could be that the *KITTI* dataset has many images where color information is fully saturated on building facades (*c.f.* Figure 4.12, bottom). In these cases, the local window of the pixel classifier is not large enough and mis-classifies Building as Sky.

In addition to labeling accuracy, we also compare the runtime of all methods. As in Section 4.4.3, we perform the runtime experiments on a machine with an NVIDIA GTX 770 GPU and Intel i7-4770 CPU. While the pixel classifier runs on GPU, we use a multi-threaded CPU implementation for Stixels. The stereo depth map is computed in advance using dedicated FPGA hardware (Gehrig et al., 2009) which introduces a delay of one frame or 50 ms, given a camera setup with 20 FPS. The comparison in Table 4.9 reveals that Stixels can be extracted faster than all reference methods, which is very important for practical applications.

In summary, we find that our extended Stixel model is faster to compute but slightly inferior in terms of labeling accuracy. However, it is important to recall here that Stixels itself, as presented in this chapter, are only the first building block for the overall automotive scene labeling concept we propose in this dissertation. Consequently,

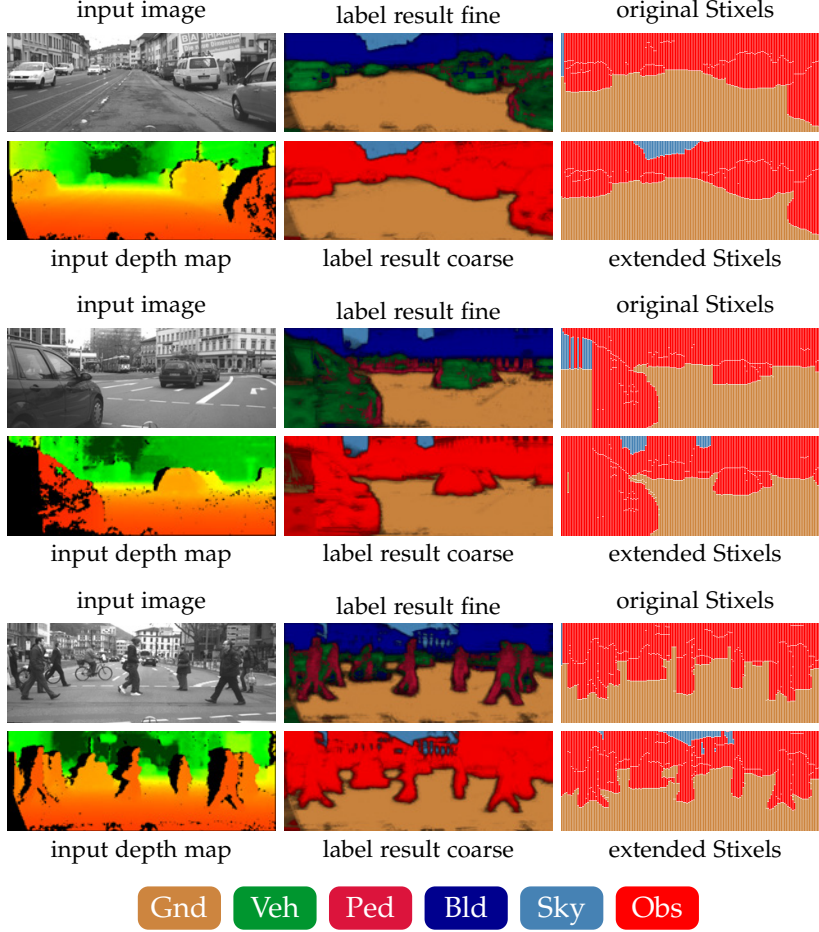


Figure 4.11: Results of our pixel-level classification approach on the *DUS* dataset and its effect when integrated into the Stixel model. We show the input and depth image (left), our pixel-level classification result for the fine and coarse label set (center), and Stixels, both in their original form as published by Pfeiffer (2011) and with our extension (right). White lines indicate a cut in the column-wise Stixel segmentation. We weight the pixel-level semantic labels (center) with the entropy of the decision to visualize the classifier’s uncertainty.

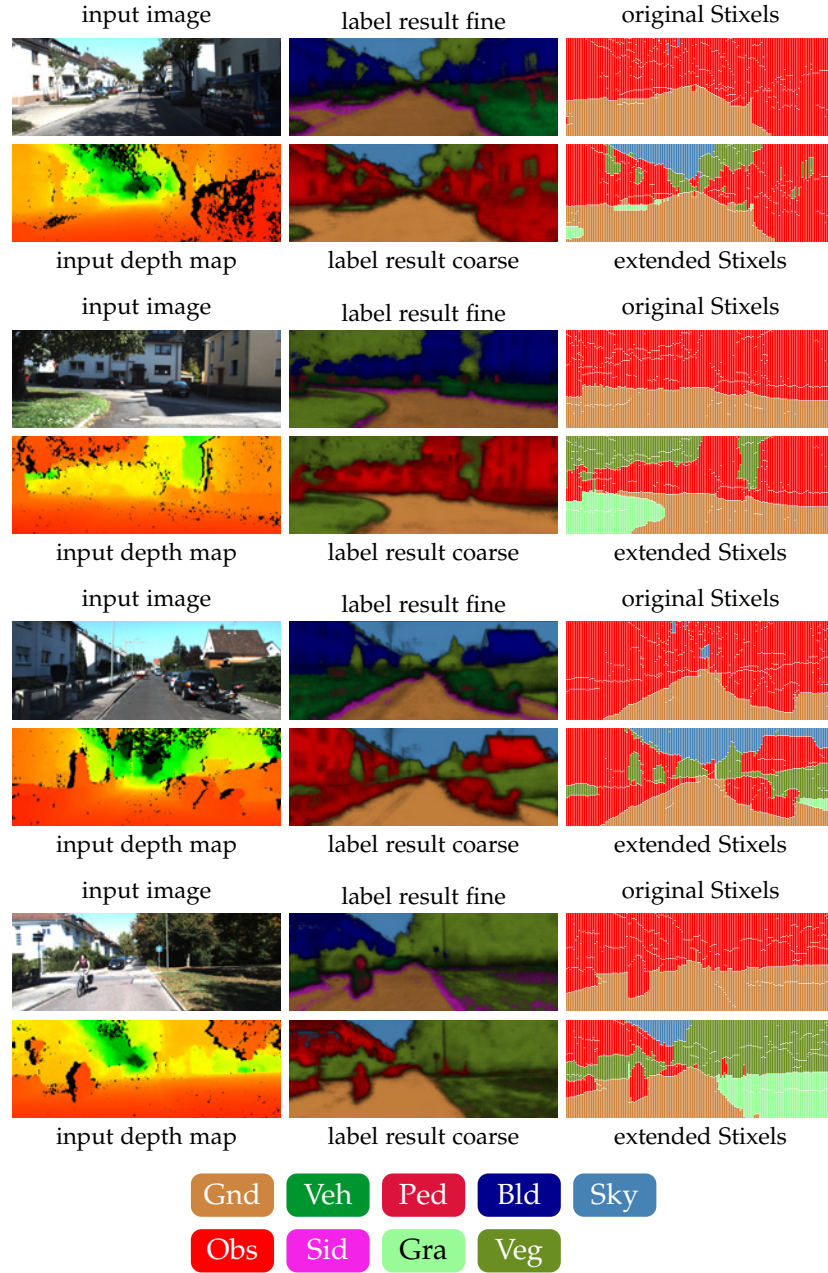


Figure 4.12: Results of our pixel-level classification approach on the *KITTI* dataset and its effect when integrated into the Stixel model. We show the input and depth image (left), our pixel-level classification result for the fine and coarse label set (center), and Stixels, both in their original form as published by Pfeiffer (2011) and with our extension (right). White lines indicate a cut in the column-wise Stixel segmentation. We weight the pixel-level semantic labels (center) with the entropy of the decision to visualize the classifier’s uncertainty.

	Gnd	Sky	Obs	Avg
Darwin unary ^{1,m}	94.5	90.6	93.1	92.7
Darwin pairwise ^{1,m}	95.8	94.2	94.7	94.9
ALE ^{2,s}	94.9	95.5	93.6	94.7
PN-RCPN ^{3,m}	96.8	91.4	95.7	94.6
Layered Int. ^{4,s}	96.4	89.5	95.1	93.7
ext. Stixels ^{5,s}	96.8	84.1	95.2	92.0

¹Gould (2012) ²Ladický et al. (2010)

³Sharma et al. (2015) ⁴Liu et al. (2015)

⁵numbers copied from Table 4.6a

^mmonocular approach ^sstereo approach

Table 4.7: Performance of our extended Stixels on the *DUS* dataset, compared to the reference methods described in Section 2.4.3.

	Gnd	Sky	Obs	Veg	Avg
Darwin unary ^{1,m}	87.0	77.5	83.9	81.1	82.4
Darwin pairwise ^{1,m}	88.9	82.7	85.8	82.5	85.0
ALE ^{2,s}	87.5	82.3	85.6	83.1	84.6
ext. Stixels ^{3,s}	89.1	71.9	79.9	71.5	78.1

¹Gould (2012) ²Ladický et al. (2010)

³numbers copied from Table 4.6b

^mmonocular approach ^sstereo approach

Table 4.8: Performance of our extended Stixels on the *KITTI* dataset, compared to the reference methods described in Section 2.4.3.

the results from this comparison must be interpreted as the first milestone towards this goal.

4.5 DISCUSSION

In this chapter, we presented a method for scene labeling that is based on classifying each individual pixel in the image. The method is composed of computing a set of local feature transformations from the color and depth image and then classifying each pixel based on a small patch around it, using a Randomized Decision Forest (RDF). We then applied this method to extend the well-known Stixel model from Pfeiffer (2011). For the sake of easier reading, we briefly recap Section 4.1, where we discussed the three main limitations of the original Stixel formulation: (1) Stixels are computed solely from depth maps, (2) they are limited to distinguish structural classes, and (3)

	<i>DUS</i>	<i>KITTI</i>
Darwin unary ¹	0.25 s	0.28 s
Darwin pairwise ¹	1.20 s	2.43 s
ALE ¹	111 s	193 s
PN-RCPN ⁴	2.8 s	<i>n/a</i>
Layered Int. ⁵	110 ms	<i>n/a</i>
ext. Stixels ^{1,2,3}	80 ms	74 ms

¹Intel i7 CPU ²GTX 770 GPU ³FPGA (for SGM stereo)

⁴Titan Black GPU, reported by Sharma et al. (2015)

⁵Tesla K40 GPU, reported by Liu et al. (2015)

Table 4.9: Runtime comparison between Stixels and the reference methods from Section 2.4.3. We measure the time required to label an image, including stereo depth map computation for Stixels (50 ms). Note that pipelining stereo and Stixel computation would allow a refresh rate of 20 FPS.

they yield a too strong (vertical) under-segmentation of the image. Our extension improves the model in all three aspects, so that it provides a better basis for subsequent algorithms compared to the original model.

However, we also discovered the limits of the local patch classification scheme. We found that it works well for certain classes, most notably for the coarse geometric and semantic layout of the scene, but it is not capable of accurately detecting all classes we are ultimately interested in. One reason for this is arguably the patch size, which is usually much smaller than actual objects in the scene. Consequently, the content of the image patch is often not very informative. However, this is only one part of the reason, as we also saw that increasing the patch size does not improve classification accuracy. The method even degrades for very large patches. This shows us that the model itself, *i.e.* the RDF classifier, is not able to extract more complex structures from the presented set of features.

As a consequence of this finding, we will now move away from treating scene labeling as a patch classification problem. In the next chapter, we will focus on methods that encode and classify larger image regions, arising from bottom-up segmentation and grouping. In this context, the extended Stixel model from this chapter will play a central role. However, the results from this chapter are not just Stixel improvements. The feature channels and the pixel-level RDF classifier will also become relevant again later in this dissertation.

Part II

REGION LEVEL

SCENE LABELING AS CLASSIFICATION OF PROPOSAL REGIONS

CONTENTS

5.1	Generation and Classification of Proposal Regions	84
5.1.1	Efficient Stixel-based Region Generation	85
5.1.2	Decision Forests for Region Encoding . .	89
5.1.3	Benefits of Depth for Encoding	91
5.1.4	Region Classification	93
5.2	Spatio-Temporal Regularization	94
5.2.1	Model Considerations	94
5.2.2	Temporal Integration	96
5.2.3	Spatial CRF Model	98
5.3	Experiments	100
5.3.1	Region Encoding	100
5.3.2	Stixel-based Regions	102
5.3.3	Spatio-temporal Regularization	105
5.3.4	Runtime	106
5.3.5	Comparison to Reference Methods	108
5.4	Discussion	109

In the previous chapter we treated scene labeling as classification of small rectangular patches around each pixel in the image and extended the Stixel algorithm to use the Stixel model assumptions as a regularizer for the pixel-level results. We found this model to be useful to quickly recover the coarse layout of the scene, but not expressive enough to accurately detect all class of interest. In this chapter, we now follow the body of literature presented in Section 2.4.1 that treats the scene labeling problem as classification of proposal regions rather than rectangular image patches. These regions typically result from bottom-up image segmentation, *i.e.* from superpixels or hierarchies of superpixels, and have the advantage that they can be of arbitrary size and shape. In this way, the image content presented to the classifier better captures actual objects in the scene, which reduces ambiguities arising from background clutter (in case of too large rectangular bounding boxes) or uninformative content (in case of too small patches). Moving from patches to regions further allows to use different methods to encode image content. While the level of information contained in small patches heavily depends on local feature transformations, larger image regions can be encoded by means

of the Bag-of-Features (BOF) approach, which better captures the co-existence of different repeating local feature patterns.

In this chapter, we present a novel way to rapidly generate proposal regions based on the extended Stixels from the previous chapter. We additionally discuss an efficient method to encode these regions and propose several possibilities to improve encoding performance in case depth information is available. Subsequently, we introduce a novel concept for spatio-temporal regularization of region-level scene labeling approaches. We will find that the presented ideas are not just an alternative solution to the approach presented in Chapter 4, but they supplement it, so that both approaches should be combined in our overall scene labeling concept. The work presented in this chapter has previously been published in (Scharwächter et al., 2013) (Section 5.1) and (Scharwächter et al., 2014) (Section 5.2).

5.1 GENERATION AND CLASSIFICATION OF PROPOSAL REGIONS¹

The region-level classification presented in the following involves two steps: generating region candidates and subsequently classifying them. Both problems will be discussed in this section, together with a proposal on how to solve them efficiently.

In Section 5.1.1, we will present an algorithm to generate proposal regions in the image by grouping adjacent Stixels based on their proximity in depth. In contrast to image segmentation methods such as SLIC (Achanta et al., 2012) or GPB-OWT-UCM (Arbeláez et al., 2011), Stixels already contain structural and semantic information about the scene. The resulting regions are thus expected to better capture actual object boundaries. At the same time, their regular layout and their properties allow for a straight-forward and efficient grouping algorithm. We will analyze the complexity of this algorithm and discuss its relation to existing segmentation and clustering methods.

Section 5.1.2 presents a tree-based method that yields a Bag-of-Features (BOF) descriptor for classification of a given image region. We briefly discussed descriptors for object detection, such as HOG and SIFT earlier in Section 1.1.2. These descriptors encode the content of a given rectangular window in the image. Using them to encode the content of an arbitrarily shaped image region however would require to define some sort of inner or outer bounding box around the region, which—depending on the shape of the region and the framing strategy—would ignore relevant content inside the region, add irrelevant content from outside the region, or both. The Bag-of-Features (BOF) approach is very suitable in this case, as it allows to only take into account local descriptors with keypoints inside the image region.

¹ Parts of this section have previously been published in Scharwächter et al. (2013) and are copied verbatim from the original publication.



Figure 5.1: Concept overview. Scene labeling as classification of bottom-up regions acquired from the Stixel representation.

In Section 5.1.3 we propose several ideas that improve region classification performance in case depth information is available. Finally, Section 5.1.4 outlines the training and inference procedure of the proposed region-level classification scheme, to provide more detailed information about the interaction between the different components discussed in this section. On a coarse level, the general steps involved in our method are depicted in Figure 5.1.

5.1.1 Efficient Stixel-based Region Generation

Following recent results that show how crucial meaningful initial regions are for segmentation performance, *e.g.* (Arbeláez et al., 2012; Carreira et al., 2012; Hu et al., 2012; Vieux et al., 2011), we present an efficient method for obtaining relevant region hypotheses by utilizing the Stixel World.

To obtain regions r_j we treat the Stixel representation as graph, as depicted in Figure 5.2 (top right). In this graph, Stixels are only connected with an edge if and only if they are spatially adjacent in the image. We start by initializing the first region r_1 at a random Stixel s_i and consecutively add neighboring Stixels to the region as long as the edge weight $w(\cdot, \cdot)$ to a *previously added* Stixel is lower than a heuristic w_{\max} . If no more Stixels are within the radius of w_{\max} , a new region is initialized at an unvisited Stixel. This process is iterated until all Stixels have been assigned to a region. The single free parameter of this algorithm is w_{\max} , which controls the size of the resulting regions. Pseudo-code is provided in Algorithm 5.1.

We choose the edge weight $w(s_n, s_k)$ between two neighboring Stixels s_n and s_k as their longitudinal distance in world coordinates, *i.e.*

$$w(s_n, s_k) = \Delta Z_{n,k} = \|Z_n - Z_k\|. \quad (5.1)$$

This choice is arguably simple, but indeed sufficient when using Stixels. Remember that all Stixels are perpendicular to the ground plane and have constant depth by definition. Furthermore, the horizontal component of their centers is equidistant in the image, so their horizontal distance is either zero (for vertical neighbors in the same image column) or constant, equal to their width (for horizontal neighbors), as apparent from Figure 5.2 (top left).

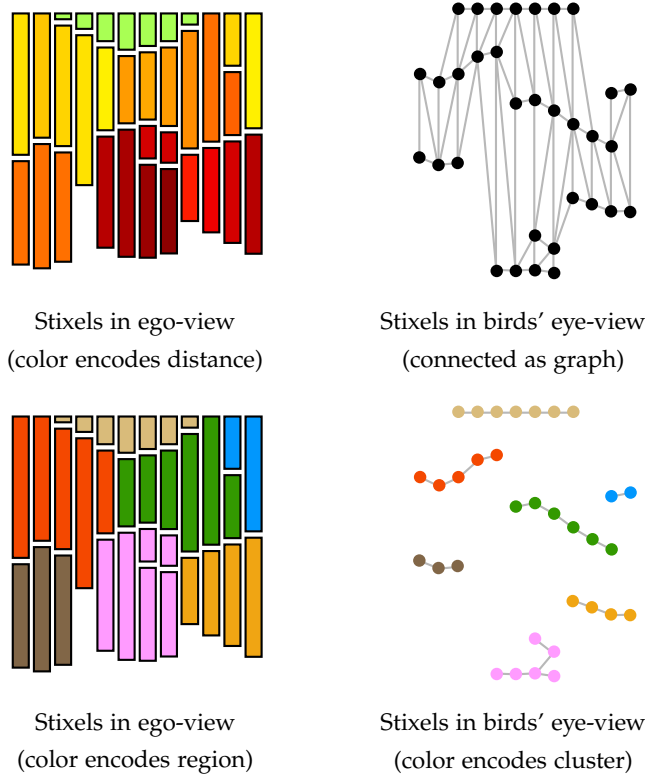


Figure 5.2: Artificial example to demonstrate the region generation process. Stixels are visualized in ego-view, as apparent in the image (top left). Stixels which are adjacent in the image are then connected in a graph, where the graph is shown in birds' eye-view, so that the length of an edge is proportional to its edge weight (top right). After applying Algorithm 5.1, we obtain clusters in this graph (bottom right) that directly correspond to regions in the image (bottom left).

After defining the edge weight, we now introduce the heuristic w_{\max} , which acts as dynamic edge weight threshold and is composed of a constant component and a dynamic component:

$$w_{\max} = \Delta Z_{\text{const}} + \sigma_Z \left(\frac{Z_n + Z_k}{2} \right), \quad (5.2)$$

where $\Delta Z_{\text{const}} \geq 0$ is a parameter and the term $\sigma_Z(Z) > 0$ captures the noise characteristics of the depth estimate.

As we assume depth is provided by a stereo camera, the uncertainty of the estimate increases quadratically with distance Z and further depends on the focal length f , stereo baseline b and the matching uncertainty σ_d of the stereo algorithm. Using the pinhole projection equation $Z = \frac{f \cdot b}{d}$, we apply error propagation to yield

$$\sigma_Z(Z) = \sigma_d \cdot \frac{Z^2}{f \cdot b}, \quad (5.3)$$

assuming that the calibration error of f and b is negligible. The constant parameter ΔZ_{const} dominates in the near field where the depth

estimate is very accurate and hence $\sigma_Z(Z)$ is small. In these cases, it can effectively avoid over-segmentation. In all other cases, the dynamic component $\sigma_Z(Z)$ dominates the heuristic and ensures that Stixels are grouped into the same region as long as their relative distance is below the distance-dependent noise level. In Figure 5.2 (bottom) we show the resulting regions and corresponding graph clusters for an artificial example. Actual Stixel-based region generation results are shown in Figure 5.3.

Algorithm 5.1 Proposal region generation from Stixels

Input: Set of Stixels \mathcal{S}

Output: Set of regions \mathcal{R}

Parameters: maximum edge weight w_{\max}

mark all Stixels as unvisited

instantiate empty queue Q

$j \leftarrow 1$

for all $s_i \in \mathcal{S}$ **do**

if s_i was visited before **then**

 continue

end if

 mark s_i as visited

 push s_i to Q

 add s_i to r_j

while Q not empty **do**

 pop s_k from Q

for all neighbors s_n of s_k **do**

if s_n was visited before **then**

 continue

end if

if $w(s_n, s_k) \leq w_{\max}$ **then**

 mark s_n as visited

 push s_n to Q

 add s_n to r_j

end if

end for

end while

$j \leftarrow j + 1$

end for

5.1.1.1 Discussion

Let us now briefly discuss some properties of the presented algorithm and its relation to other clustering and image segmentation algorithms. Note that the two problems of clustering and image segmentation are very different problems in general, but they overlap in the area of *graph clustering*, where the goal is to find clusters of similar

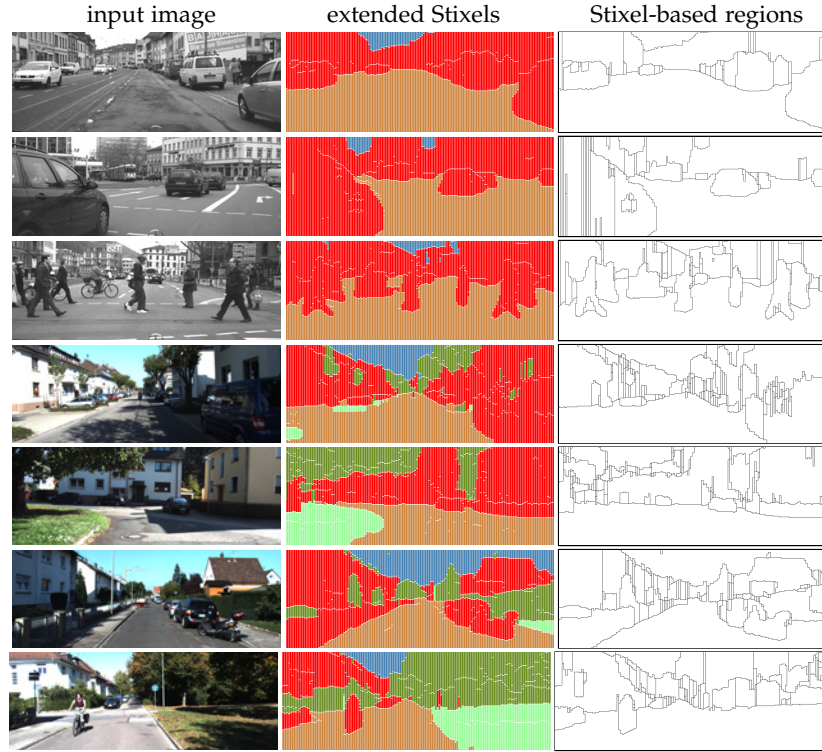


Figure 5.3: Stixel-based region results on the *DUS* and *KITTI* dataset. Stixels are grouped according to Algorithm 5.1. It can be seen that most relevant objects in the scene are captured by one distinct image region, as long as an object can be separated from its background in the longitudinal direction. On the other hand, noisy depth estimates, e.g. due to reflections on vehicles, can yield an over-segmentation of objects.

nodes which are connected via edges in a graph. In the case of image segmentation, nodes correspond to pixels or superpixels, while edges describe their spatial neighborhood in the image. As a result, clusters in the graph directly correspond to segments or regions in the image. Several image segmentation algorithms (e.g. Felzenszwalb and Huttenlocher (2004); Ladický et al. (2013); Vicente et al. (2008)), including our region generation algorithm from above, follow this principle, which is why we use both terms interchangeably at this point.

Algorithm 5.1 directly implements the idea of connected component analysis in graphs. It is also strongly related to image processing algorithms for blob extraction and flood filling. The appeal of this algorithm is its simplicity and fast execution. Due to the regular graph structure, where every Stixel only has about five connected neighbors, the runtime complexity of the algorithm is linear in the number of Stixels. Furthermore, the solution of the algorithm does not depend on the choice of the initial seed point, as long as edge weight $w(\cdot, \cdot)$ and heuristic w_{\max} between two Stixels are symmetric.

As a final remark, our goal is to rapidly obtain reasonably good candidate regions and the greedy approach we propose to use certainly introduces errors in terms of over and under-segmentation. However, we are partially able to recover from those errors by means of spatio-temporal regularization, which will be presented later in Section 5.2. Note that this is in line with our general approach of decomposing the scene labeling problem, as presented in Section 1.2.3: Finding practical solutions for subproblems and at the same time combining the results in a way that errors of different building blocks are not correlated and can thus be resolved later.

5.1.2 Decision Forests for Region Encoding

Now that we have introduced a method to efficiently *generate* image regions, we discuss an approach to efficiently *encode* the content of a given region. The encoding step yields a region-level descriptor that can be used to predict a semantic class label for each region.

In Section 2.3.1 we briefly introduced different feature encoding methods that implement the Bag-of-Features (BOF) paradigm. We already discussed that tree-based methods, in particular the Extremely Randomized Clustering (ERC) forests proposed by Moosmann et al. (2008), are superior to standard k-Means vector quantization, both in terms of runtime and accuracy. Therefore, we adopt the ERC-Forest approach in this section and provide more details about the way we apply it in our scene labeling concept. Note that ERC-Forests and RDFs are structurally identical, so that the terms can be used interchangeably. However, the ERC terminology helps to highlight the fact that the principal use case of the decision tree is not classification but clustering of the underlying feature space.

The general principle of RDFs for classification has been discussed in Section 4.2.2.1 already. One aspect we did not mention at that point however is that the learned structure of a decision tree is equivalent to a partitioning of the underlying feature space. While this is not surprising, and holds true for all classification methods in general, decision trees make this partitioning very explicit, as each leaf node corresponds to a unique partition in space. Consequently, the tree can be used as a codebook that maps a high-dimensional feature descriptor to a unique index. This concept is visualized in Figure 5.4. In our case, where the decision tree is trained in a supervised fashion, these partitions directly correspond to clusters of training samples with one specific class label. The resulting supervised clusters are therefore also potentially more informative for feature encoding than the unsupervised clusters found by k-Means. One downside of this approach is that encoding a feature with a *single* tree is not as informative as an encoding based on k-Means. This is because a tree inspects only few feature dimensions, while a k-Means cluster cen-

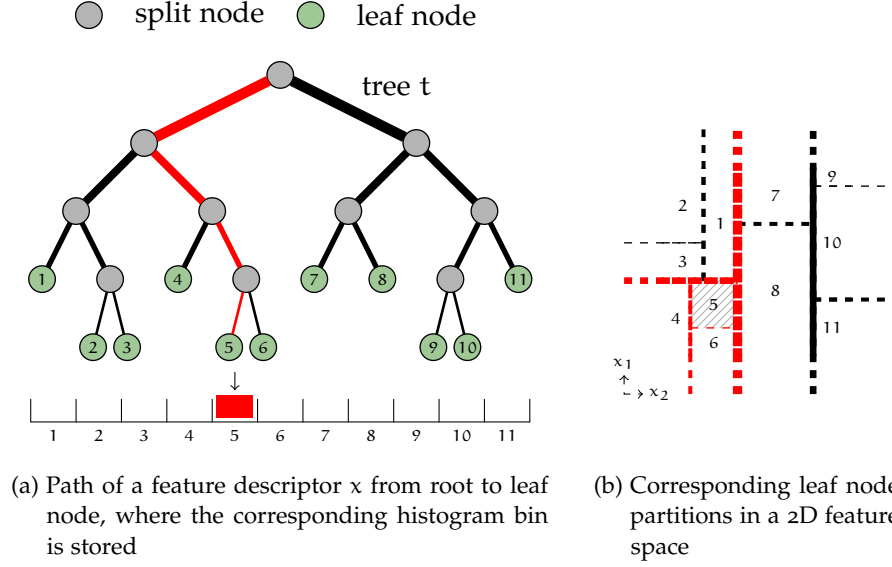


Figure 5.4: Example of a decision tree applied to feature encoding. A feature descriptor x traverses the tree from root to leaf node. Each leaf node has a unique ID that points to the corresponding bin in the histogram. The right side shows a two-dimensional example of the resulting partitioning in feature space, where the line thickness correlates with tree depth and the red lines are split functions evaluated along the path to leaf number 5.

ter takes into account all dimensions. However, as with classification, moving from single trees to ensembles of trees improves performance considerably (Moosmann et al., 2008).

To not sacrifice encoding accuracy, we follow a large body of literature and use SIFT (Lowe, 2004) as local feature descriptor. We extract SIFT features on multiple scales and on a dense grid over the image. An encoding RBF is then trained in the standard supervised fashion presented in Section 4.2.2.2. In other words, the RBF is trained to classify an image patch based on the corresponding SIFT descriptor. To encode a larger region R_i , we collect all SIFT descriptors d_j , where the descriptor keypoint k_j lies within the region R_i . We then use the RBF to vector-quantize each descriptor d_j to $\text{idx}(d_j)$ and spatially pool all indices $\text{idx}(d_j)$ over R_i into a histogram. To build the ensemble histogram, we concatenate all tree-wise histograms, so that for T trees and L leaf nodes per tree, the resulting BOF histogram is of size $H_{\text{ERC}} = T \cdot L$.

To have full control over the histogram length, we first grow each tree to full depth and then prune it back bottom up. Candidates for pruning are split nodes where both child nodes are leaves. In each pruning iteration, we randomly sample a node from the list of all pruning candidates and prune its children, so that the split node becomes a new leaf. This process is continued until the desired number of leaf nodes is reached.

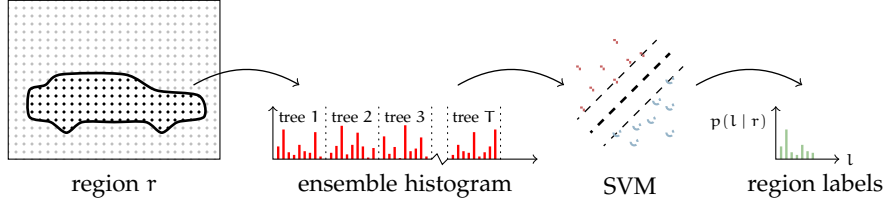


Figure 5.5: Steps involved to encode a region r : All descriptors within region r traverse the trees and a histogram of leaf node indices is accumulated. This histogram is then classified with a Support Vector Machine (SVM) to obtain a multi-label classification response. Descriptors are extracted only once on a dense grid over the whole image, indicated by gray dots. However, only descriptors with keypoints inside of region r are considered to encode this particular region. The corresponding keypoints are printed in black.

Based on the resulting region-level histogram, we train a Support Vector Machine (SVM) to obtain a multi-label classification response for each region. More details on the training procedure and the SVM classifier will be discussed later in Section 5.1.4. The single steps of region encoding and classification are depicted in Figure 5.5.

5.1.3 Benefits of Depth for Encoding

Given the success of multi-cue methods for geometrically rigid object classification, *e.g.* (Keller et al., 2011), which combine grayscale imagery with other modalities, such as dense stereo or optical flow, little work has been done on transferring those ideas to Bag-of-Features (BOF) methods. In this section, we fill this gap and discuss three possibilities of leveraging dense stereo information in different processing stages of a BOF pipeline. In particular, we are going to discuss (1) *descriptor scale selection* to alleviate the need of extracting features at multiple image scales, (2) *bag-of-depth-features* to classify regions based on repeating depth patterns, and (3) *height pooling* to introduce a 3D geometric ordering into the region descriptor.

DESCRIPTOR SCALE SELECTION The inherently unknown scale of objects in an image is a common problem in many computer vision tasks. A widespread solution to this problem is to extract features at multiple scales (Arbeláez et al., 2012; Carreira et al., 2012; Sturges et al., 2009) and we also followed this approach earlier in Section 5.1.2. However, in terms of computational costs, choosing the single *correct* scale is arguably more efficient. With dense stereo at hand, this problem can be alleviated by extracting features only at a single scale, which is dynamically derived from the 3D position of a keypoint. For each descriptor keypoint, we triangulate the corresponding 3D location, center a bounding box that covers a fixed pre-defined area at

this location (e.g. $0.5\text{m} \times 0.5\text{m}$) and project the borders of the bounding box back to the image. The descriptor is then extracted from the resulting image patch. In our experiments in Section 5.3.1, we compare this strategy against the most common approaches of choosing a single fixed scale and extracting descriptors at multiple scales.

BAG-OF-DEPTH-FEATURES The high density of depth images computed with SGM allows us to extract complex features (such as SIFT) directly on the depth image, similar to (Keller et al., 2011). We implement this idea by following the same BOF pipeline as described in Section 5.1.2, where we replace the grayscale image with the dense depth image. The resulting Bag-of-Depth-Features (BODF) histogram then encodes the distribution of typical depth patterns within an image region. In our experiments in Section 5.3.1, we evaluate the performance of this approach against the classical BOF pipeline, as well as a combination of both approaches, which is obtained by concatenating the BOF and BODF histograms before SVM classification.

HEIGHT POOLING The spatial pyramid representation of Lazebnik et al. (2006) is commonly used to overcome the prevalent loss of geometric ordering in a classical BOF approach, c.f. Section 2.3.2. The concept of *height pooling* that we propose here is a 3D-variant of this spatial pyramid representation, where visual words are pooled into different histograms according to their height above the ground plane to incorporate a vertical geometric ordering into the region descriptor.

Instead of pooling all descriptors of a region into one BOF histogram of length H_{ERC} , we define B discrete height bands in 3D world coordinates. Each such band has its own histogram and each descriptor is pooled into the corresponding band-specific histogram according to the 3D vertical position of its keypoint. The final region-level descriptor is subsequently formed by concatenating all band-specific histograms. Consequently, the BOF histogram after height pooling has $H_{\text{HP}} = B \cdot H_{\text{ERC}}$ dimensions. To obtain the height above the ground plane for a keypoint, we use the component Y_w in Equation (4.3), as described earlier in Section 4.2.1.3. De-

scriptors without valid depth measurement are always pooled into the histogram of the last height band. The key property of the resulting region descriptor is that it takes into account 3D vertical structure of objects but is largely invariant to the 3D lateral and longitudinal position of an object w.r.t. to the ego-vehicle, which is exactly what

Band	Height [m]		
1	$-\infty$	–	0.0
2	0.0	–	0.5
3	0.5	–	1.0
4	1.0	–	1.5
5	1.5	–	2.0
6	2.0	–	4.0
7	4.0	–	10.0
8	10.0	–	∞

Table 5.1: List of chosen height bands.

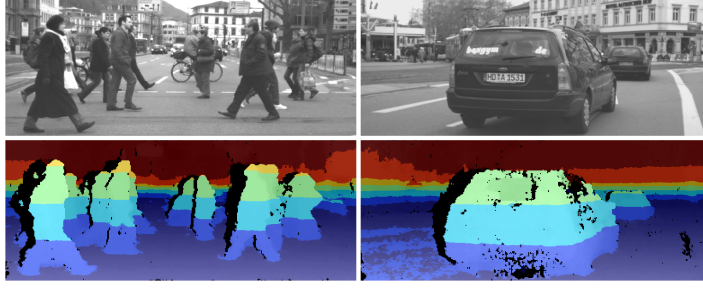


Figure 5.6: Illustration of the height bands we use for pooling. We show two camera images and the corresponding disparity images overlaid with the color coded height bands from Table 5.1. Each descriptor is pooled into the band-specific histogram according to the height above the ground plane of its keypoint.

is needed for street scenes, where the same type of object can occur at various lateral positions and distances, while the vertical structure and scale of objects in one category is largely consistent in the 3D world. In our experiments we use $B = 8$ height bands, empirically chosen to obtain good coverage for the object classes of interest. More precisely, we pool encoded descriptors into the height bands given in Table 5.1. Figure 5.6 illustrates these height bands on two real-world examples. In Section 5.3.1 we evaluate this extension against the orderless ERC region histogram.

It should be noted that a similar idea has been presented by Gupta et al. (2013). They propose histograms of *geocentric textons*, which they describe as “vector quantized words in the joint two-dimensional space of height from the ground and local angle with the gravity direction”.

5.1.4 Region Classification

As classifier for our region-level histogram descriptor we use an SVM with Histogram Intersection Kernel (HIK), presented by Wu (2010). For our problem, the HIK-SVM showed to be superior in terms of predictive accuracy *and* execution time, compared to both linear SVMs and SVMs with Radial Basis Function (RBF) kernel. It also showed to be more robust under fewer samples compared to a Randomized Decision Forest (RDF) classifier. To obtain a multi-class response from the classifier, we follow a one-vs-all approach, *i.e.* we train a two-class SVM model for each class that separates it from all other classes. To combine the results of all classifiers, we perform a sigmoid mapping of the individual SVM scores and normalize their sum to one. Note that this normalization is not necessary for the pure classification task, but it is required for soft class probability estimates, which will become important later in Section 5.2.2.

5.2 SPATIO-TEMPORAL REGULARIZATION¹

With the concepts presented in Section 5.1 we now have an alternative region-level solution to the scene labeling problem, in addition to the pixel-level classifier from Section 4.2. In Section 4.3, we extended the Stixel model and could show that Stixels spatially regularize the pixel-level classification results, thereby improving labeling performance. In this section, we also focus on regularization, but present a method that is better suited for region-level models.

In addition to spatial regularization, we now focus stronger on regularization of class labels over time. While a solution to temporally consistent scene labeling is interesting and desirable in general, there is in fact a good reason why we tackle this problem now, on the region level, rather than on the pixel level: We observed that enforcing temporal consistency on the pixel level is computationally *more* demanding and at the same time *less* important, while enforcing it on the superpixel or region level is *less* demanding and at the same time *more* important. The computational argument can easily be explained with the number of temporal correspondences that have to be found and processed. The higher importance of temporal regularization on the region level has to do with the number of samples that are available for training the classifiers of the two individual levels. On the pixel level, we train a classifier on local image patches. These patches have limited information content, but we have many samples for training. As a result, the classifier cannot estimate all labels with high accuracy, but the results are very consistent over time, because the classifier can learn a very smooth model from the large amount of samples and can thus generalize better. On the region level, it is the other way around: An image region has more information content, but we have fewer samples for training. Consequently, we have a classifier with higher per-label precision, but the generalization is not as good, which leads to occasional mis-classifications over time. Additionally, the Stixel-based region generation process itself is not error-free, which further amplifies this problem. As a solution, we will introduce our concept for spatio-temporal regularization of region-level classification results in the next sections, together with a discussion about the design choices that led to this concept.

A large number of samples is crucial to train a good classifier model.

5.2.1 Model Considerations

Before we put forward our concept for spatio-temporal regularization of a semantic scene labeling result, let us briefly recapitulate Section 2.4.2, where we discussed and categorized different approaches of combining information from multiple time steps. In Section 2.4.2,

¹ Parts of this section have previously been published in (Scharwächter et al., 2014) and are copied verbatim from the original publication.

we differentiated *batch* or *offline* methods that are suitable for post-processing a sequence by taking into account the entire video, and *causal* or *online* methods that only take into account past measurements. In an automotive setup, temporal regularization has to be performed in a live video stream, which is why our approach should fall into the latter category of causal processing.

With Stixels being the finest elements that describe a scene, our approach is also related to the superpixel-based approaches that we discussed earlier in Section 2.4.2. In fact, there is a large body of literature promoting spatial regularization of class labels on superpixel level to remove spurious mis-classifications. However, when it comes to the temporal domain, most approaches focus on improving the temporal consistency of superpixel boundaries, rather than focusing on the regularization of their class label. As we are using Stixels, we cannot follow this strategy, as Stixels are bound to image columns with a fixed width by design. Also, having consistent superpixel boundaries over time still does not yield temporally consistent labels. It just improves temporal correspondence.

With this in mind, we raise the question on which level temporal consistency of labels should ideally be tackled. We argue that proper *unique* temporal correspondence is the key to success, as it allows to use well-established mathematical models for label regularization, such as Bayesian recursive estimation. Regularization on the dense pixel level requires dense optical flow, which is often overly smooth in regions with low image texture and can thus lead to false correspondences. It is also—despite modern parallel GPU implementations—computationally expensive. Regularization on the superpixel level on the other hand requires superpixels with temporally consistent boundaries to provide unique correspondences. As discussed before, this is not possible with Stixels.

We therefore decide to perform temporal regularization on the level of sparse point trajectories. Tracking of sparse well-textured image corners over time is a classical computer vision problem that can be solved with high accuracy, robustness, and efficiency. Such a tracker is able to provide unique correspondences, meaning that the object behind a single tracked point should in fact never change during the lifetime of the track, *i.e.* the only cause of a label change is misclassification. Another practical argument is that tracking of sparse points is typically performed in our overall automotive vision architecture anyway, *e.g.* for the 6D-Vision approach by Franke et al. (2005). This means that point tracks are available without any additional computational overhead. Our decision is further supported by Ellis and Zografos (2012), where the combination of sparse point trajectories together with superpixels results in more efficient models with adequate performance. This decision also implies another important design choice: To model spatial and temporal regularization indepen-

dently. This might sound suboptimal at first glance, but it again has practical benefits. Most importantly, it allows better pipelining of all components, which allows better use of available resources.

In summary, we model spatial and temporal regularization independently: For consistency in the temporal domain, we opt for a Bayesian filtering scheme on the level of sparse long-term point trajectory, which effectively combines the efficiency of recursive methods (Erbs et al., 2012; Ess et al., 2009; Wojek and Schiele, 2008; Wojek et al., 2013) with the robustness of considering a longer temporal window (Floros and Leibe, 2012; de Nijs et al., 2012). For consistency in the spatial domain, we instead model a Conditional Random Field (CRF) on the level of Stixels. The order in which spatial and temporal regularization is applied is in fact not overly important. Both methods require class label distributions as input in order to regularize successfully. For practical reasons however, we decide to first filter semantic labels over time and subsequently perform spatial CRF regularization on the Stixel level, because it avoids the extra effort of computing CRF marginals. Without marginals the CRF only yields the most likely class label, but no scores that tell how certain this decision is, which would however be needed as input for temporal regularization. Accordingly, we first describe our method to filter the semantic label decision over time in Section 5.2.2 and present the spatial CRF model in Section 5.2.3.

5.2.2 Temporal Integration

In contrast to existing methods discussed in Section 2.4.2, which aim to propagate the semantic label decision either densely on the pixel level (Miksik et al., 2013) or focus explicitly on registering superpixels over time (Couprie et al., 2013), we regularize labels over time locally on sparse long-term point trajectories, where correspondence is very reliable and can be computed efficiently. For this we use an implementation of the well-established Kanade-Lucas-Tomasi (KLT) feature tracker of Tomasi and Kanade (1991) that has been developed for the 6D-Vision approach by Franke et al. (2005).

Regularization is performed using Bayesian recursive estimation. It should be pointed out again that this concept is completely independent of the level it is applied on. We use it on sparse trajectory level in our overall system concept, but it can equally well be applied on the dense pixel level or superpixel level, as long as there exists a history of unique temporal correspondences. In the following, we will formally define the problem of Bayesian recursive estimation for discrete class labels and present some insights into its behavior by simulation with artificial input data.

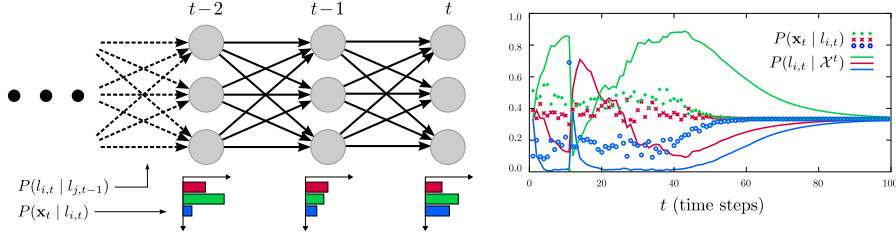


Figure 5.7: Illustration of the recursive label filtering scheme, which is applied to each point trajectory. Unfolded directed Markov model with three labels (left). Arrows indicate possible causal transitions. Simulated result for a trajectory of length 100 (time steps), where dots represent noisy observations and solid lines show the resulting filtered posterior for the labels in each time step, given all previously observed data (right). In time step 10, we place a strong outlier in the observation and starting from time step 40, we quickly shift all observations towards a uniform value of $P(\mathbf{x}_t | \mathbf{l}_{i,t}) = \frac{1}{3} \forall i$ to demonstrate the filtering effect of the model. Note that in practice we never observed such strong outliers as simulated in time step 10. The weight α is set to $\alpha = 0.95$, as in all our experiments.

BAYESIAN RECURSIVE ESTIMATION To integrate the semantic region classification output from Section 5.1.2 over time, we opt for a Hidden Markov Model (HMM) approach. We model label transitions as a Markov chain for each trajectory and solve a discrete filtering problem in the Bayesian sense, as shown in Figure 5.7. For a trajectory with an age of t time steps, we estimate the posterior $P(\mathbf{l}_{i,t} | \mathbb{X}^t)$ of label $\mathbf{l}_{i,t}$, given the set of all previous and current observations $\mathbb{X}^t = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_t\}$ up to time t . Prediction is performed using forward inference, which involves recursive application of predict (5.4) and update (5.5) steps (see e.g. Murphy (2012)):

$$P(\mathbf{l}_{i,t} | \mathbb{X}^{t-1}) = \sum_j P(\mathbf{l}_{i,t} | \mathbf{l}_{j,t-1}) P(\mathbf{l}_{j,t-1} | \mathbb{X}^{t-1}) \quad (5.4)$$

$$P(\mathbf{l}_{i,t} | \mathbb{X}^t) = \frac{P(\mathbf{x}_t | \mathbf{l}_{i,t}) P(\mathbf{l}_{i,t} | \mathbb{X}^{t-1})}{\sum_j P(\mathbf{x}_t | \mathbf{l}_{j,t}) P(\mathbf{l}_{j,t} | \mathbb{X}^{t-1})}. \quad (5.5)$$

The term $P(\mathbf{l}_{i,t} | \mathbf{l}_{j,t-1})$ in (5.4) corresponds to the transition model of labels between two subsequent time steps and acts as the temporal regularizer in our setup. Note that ideally objects do not change their label over time, especially not on the trajectory level. The only two causes for a label change are errors in the observation model $P(\mathbf{x}_t | \mathbf{l}_{i,t})$ or measurement errors in the trajectory, *i.e.* the tracked point is accidentally assigned to another object. Thus, we assign a relatively large weight $\alpha \in (0, 1)$ to the diagonal entries of the transition

matrix (self loops) and a small value to the remaining entries, such that we obtain a normalized probability distribution:

$$P(l_{i,t} | l_{j,t-1}) = \begin{cases} \alpha & i = j \\ \frac{1-\alpha}{L-1} & i \neq j. \end{cases} \quad (5.6)$$

We empirically choose $\alpha = 0.95$ in our experiments. Alternatively, the transition probabilities could be learned from training data.

Following Section 5.1.2, we model the observation \mathbf{x}_t as the BOF histogram $\mathbf{h}(R_k)$ of the region R_k covering the tracked feature point in time step t . From Section 5.1.4 we get the per-label classification output $\Gamma_i(\mathbf{h}(R_k)) \in (0, 1)$, with $i = 1 \dots L$ and L denoting the number of labels. We relate this score to the observation model given uniform label priors as $P(\mathbf{x}_t | l_{i,t}) \propto \Gamma_i(\mathbf{h}(R_k))$.

To maintain a constant number of tracks, new trajectories are instantiated in every time step to account for lost tracks. The label prior of a new trajectory is chosen uniformly as $P(l_{i,t=0} | \emptyset) = \frac{1}{L}$. In Figure 5.7, we illustrate the recursive label filtering process and provide a simulation for better insight into the model behavior. To assign the track-level filtered label posteriors back to proposal regions, we compute the average posterior over all trajectories α within region R_k , where $A(R_k)$ is the total number of trajectories in the region, *i.e.*

$$\bar{P}(l_i) = \frac{1}{A(R_k)} \sum_{\alpha=1}^{A(R_k)} P_{\alpha}(l_{i,t} | \mathbf{X}^t). \quad (5.7)$$

5.2.3 Spatial CRF Model

One side effect of most data-driven grouping schemes, such as the one presented in Section 5.1.1, is local spatial over-segmentation. To incorporate global smoothness properties, we formulate a Conditional Random Field (CRF) in each time step, where Stixels are connected within a graphical model.

More formally, a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ consisting of vertices \mathcal{V} and edges \mathcal{E} is built, where $|\mathcal{V}| = N$ is the number of vertices (Stixels). We assign a set of discrete random variables $\mathcal{Y} = \{y_n | n = 1 \dots N\}$, where each y_n can take a value of the label set $\mathcal{L} = \{l_i | i = 1 \dots L\}$. A labeling $\mathbf{y} \in \mathcal{L}^N$ defines a joint configuration of all random variables assigned to a specific label. In a CRF, the labeling \mathbf{y} is globally conditioned on all observed data \mathbf{X} and follows a Gibbs distribution:

$$p(\mathbf{y} | \mathbf{X}) = \frac{1}{Z} \exp \left(- \sum_{c \in \mathcal{C}} \psi_c(\mathbf{y}_c) \right), \quad (5.8)$$

where Z is a normalizing constant, \mathcal{C} is the set of maximal cliques, \mathbf{y}_c denotes all random variables in clique c and $\psi_c(\mathbf{y}_c)$ are potential

functions for each clique (Koller and Friedman, 2009), having the data \mathbf{X} as implicit dependency. Finding the Maximum A-Posteriori (MAP) labeling $\hat{\mathbf{y}}$ is equivalent to finding the corresponding minimum Gibbs energy, *i.e.*

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} p(\mathbf{y} | \mathbf{X}) = \arg \min_{\mathbf{y}} E(\mathbf{y}) . \quad (5.9)$$

For semantic labeling problems, the energy function $E(\mathbf{y})$ typically consists of unary (ψ_n) and pairwise (ψ_{nm}) potentials and is defined as

$$E(\mathbf{y}) = \sum_{n \in \mathcal{V}} \psi_n(y_n) + \sum_{(n,m) \in \mathcal{E}} \psi_{nm}(y_n, y_m) . \quad (5.10)$$

In the following, we discuss the potential functions ψ_n and ψ_{nm} in more detail.

DATA TERM For the unary potential of a vertex (*i.e.* a Stixel), we employ the filtered track-level posteriors from Equation (5.7) that have been averaged over the corresponding proposal region of a Stixel, *i.e.*

$$\psi_n(y_n = l_i) = -\log(\bar{P}(l_i)) . \quad (5.11)$$

Note that this unary potential not only incorporates data from a single Stixel locally but from the larger proposal region it is part of. In CRFs (compared to MRFs) the labeling \mathbf{y} is globally conditioned on the data \mathbf{X} , so this is a valid choice to increase the robustness of the unary term.

SMOOTHNESS TERM To encourage neighboring Stixels to adopt the same label, the pairwise potentials take the form of a modified Potts model. In pixel-level CRFs, the Potts model is often extended to be contrast-sensitive to disable smoothing when strong changes in image intensities occur, *e.g.* (Sturges et al., 2009). In the same spirit, our Stixel-level model allows us to make smoothing sensitive to similarity measures derived from 2D geometry, 3D geometry, or even 3D motion direction of Stixels. While all of these examples are viable options, we do not investigate too much in this direction and instead propose a measure of *common boundary length* between neighboring Stixels to model their similarity. Hence, we define the pairwise potentials as

$$\psi_{nm}(y_n, y_m) = \begin{cases} 0 & y_n = y_m \\ \gamma \Omega(n, m) & y_n \neq y_m, \end{cases} \quad (5.12)$$

where γ is a smoothness factor. The term $\Omega(n, m)$ measures the similarity of adjacent Stixels. For two adjacent Stixels n and m , let b_n be the number of pixels on the boundary of Stixel n . Further let c_{nm} be the number of pixels on the boundary of Stixel n , which are direct

neighbors of Stixel m . The terms b_m and c_{mn} are defined accordingly. The common boundary length is then defined as

$$\Omega(n, m) = \frac{c_{nm} + c_{mn}}{b_n - c_{nm} + b_m - c_{mn}}. \quad (5.13)$$

By definition, the measure is symmetric and limited to the range $[0, 1)$ for non-overlapping Stixels and the cost of a label change is reduced if two adjacent Stixels only have a small common boundary.

5.3 EXPERIMENTS

The experimental section of this chapter covers five different aspects. We start by evaluating the proposed region encoding and classification concept in Section 5.3.1. To not bias these results by potential errors in the region *generation* process, we perform these first experiments on ground truth regions. The different methods to incorporate depth information into the encoding step (*c.f.* Section 5.1.3) are also evaluated in this part. In Section 5.3.2 and Section 5.3.3, we then take a look at actual scene labeling results using our Stixel-based regions and also evaluate the proposed spatio-temporal regularization. In Section 5.3.4 and Section 5.3.5 we finally analyze the runtime of our algorithm and compare it against the reference methods from Section 2.4.3.

5.3.1 Region Encoding

To evaluate the accuracy of region encoding and classification independently of the region generation process, we first assume ground truth regions to be provided by an oracle. Table 5.2 shows the results for different variants of this experiment, using the ERC encoding method with SIFT descriptors at multiple scales (prefix “M/SIFT-”) and Haar descriptors (Stollnitz et al., 1995) at a single fixed scale (prefix “S/Haar”). The superscript “HP” indicates the use of height pooling, as proposed in Section 5.1.3, and will be discussed later. The subscript “G” in ERC_G marks that descriptors are extracted from the grayscale image. We use the publicly available VLFEAT library (Vedaldi and Fulkerson, 2008) to extract SIFT descriptors. For the ERC forest, we train a model with $\mathcal{T} = 5$ trees that have $\mathcal{L} = 500$ leaf nodes each. As reference, we also provide a region-level classifier variant based on averaging the pixel-level classifier from Chapter 4 over each ground truth region (last row in Table 5.2).

It is interesting to see that the variant based on our pixel-level classifier yields better classification accuracy for the coarse structural classes, *i.e.* ground and sky, while the region-level ERC_G method from this chapter is better in predicting the remaining classes. We take this as another evidence that it is meaningful to use pixel-level semantics early in the overall system, *i.e.* to extend Stixels with semantics,

as already demonstrated in Section 4.4.4. This finding also promotes the idea of a hierarchical classification approach, where the coarse semantic levels are detected by means of pixel-level classification and Stixels, and only the more fine-grained levels of semantics that split up generic obstacles into classes such as pedestrians, vehicles and buildings are resolved with a region-level approach. Following this strategy would increase accuracy and reduce computational burden at the same time by focusing on segments that cannot be classified accurately on the pixel level. We will follow up on that idea again later in Section 6.2. In addition to using SIFT descriptors at multiple scales, as discussed in Section 5.1.2, we perform the same experiment again with single-scale Haar descriptors, which are much faster to compute compared to SIFT. The trade-off between runtime and accuracy will be discussed in more detail later in Section 5.3.4.

DESCRIPTOR SCALE SELECTION Extracting SIFT descriptors at multiple scales entails a high computational effort. We therefore compare the multi-scale variant against single-scale variants, where the scale is (1) kept fixed, and (2) computed automatically from the depth data. The latter approach corresponds to the proposed descriptor scale selection. Figure 5.8 (left) shows results when using only a single scale, where the cell size of SIFT is varied (the descriptor is built from 4×4 cells, as proposed by Lowe (2004)). In Figure 5.8 (right), the best performance with a fixed single scale (cell size 4 px) is compared against a multi-scale approach using the six best performing single scales (omitting cell sizes of 48 px and 64 px). Furthermore, we show the performance when using a single scale, where the cell size is adapted dynamically (using depth information) to cover a fixed width of 0.75 m in metric 3D space. Region classification accuracy is shown by means of ROC curves for the Vehicle class, because it better visualizes small differences in performance compared to the IOU score. We observe that dynamic scale selection helps significantly when only a single

	Gnd	Veh	Ped	Sky	Bld	Avg	Dyn
M/SIFT-ERC _G	88.8	87.7	78.6	94.8	85.1	87.0	83.2
M/SIFT-ERC _G ^{HP}	95.0	88.4	82.6	92.9	94.2	90.6	85.5
S/Haar-ERC _G	91.4	82.6	75.3	98.3	84.0	86.3	79.0
S/Haar-ERC _G ^{HP}	90.8	86.0	80.9	95.5	93.6	89.4	83.5
Avg. Pixel Classifier	92.1	63.9	74.8	100.0	73.5	80.8	69.3

Table 5.2: Results of ERC_G region encoding and classification on the DUS dataset in case ground truth regions are provided by an oracle. As metric we use the IOU score. The superscript “HP” indicates use of height pooling.

scale is used. However, using features at multiple scales still yields better results.

BAG-OF-DEPTH-FEATURES We denote our proposed Bag-of-Depth-Features (**BODF**) variant as ERC_D (descriptors extracted on depth images) and compared it against the baseline encoding method ERC_G (descriptors extracted on grayscale images), as well as the O_2P region descriptor by Carreira et al. (2012). We adapted their publicly available code to our *DUS* dataset and use their best performing single descriptor eMSIFT-F with $\log(2\text{AvgP})$ pooling, omitting the dimensions for color. The results in Figure 5.9 indicate that ERC_D is outperformed by the grayscale variant ERC_G , while the combination ERC_{GD} , where both descriptors are concatenated, is better than the individual descriptors on average. O_2P outperforms the **ERC** methods in this first experiment on ground truth regions. However, in later experiments on inferred regions this is not the case anymore.

HEIGHT POOLING In almost all experiments we conducted, the proposed height pooling extension (superscript “HP”) increased region classification accuracy, *c.f.* Table 5.2, Figure 5.9, and Table 5.4. The multi-cue grayscale and depth descriptor (ERC_{GD}) outperforms the individual variants (ERC_G and ERC_D) and height pooling results in a significant performance gain (ERC_G^{HP} *vs.* ERC_G and $\text{ERC}_{GD}^{\text{HP}}$ *vs.* ERC_{GD}). However, these observed improvements do not add-up. Integrating both multiple cues and height pooling in a single descriptor ($\text{ERC}_{GD}^{\text{HP}}$) does not improve over using height pooling (ERC_G^{HP}) alone.

5.3.2 Stixel-based Regions

In a similar spirit to the previous experiment, we now evaluate the performance impact of using Stixels as regions in an artificial setup.

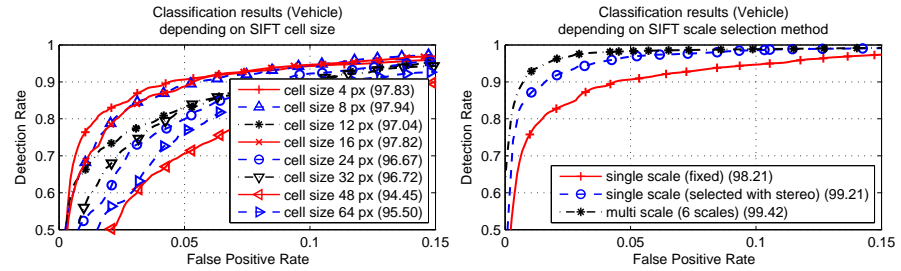


Figure 5.8: Region classification performance on the *DUS* dataset when using a single **SIFT** scale with *fixed* cell size (left). Best result using a single fixed scale, a single scale selected with stereo and multi-scale using six scales (right). Additionally, the Area Under Curve (**AUC**) (in %) is given in the legend. Note that (right) contains the boundary **ROC** over all single scales from (left), which explains the slightly increased **AUC**.

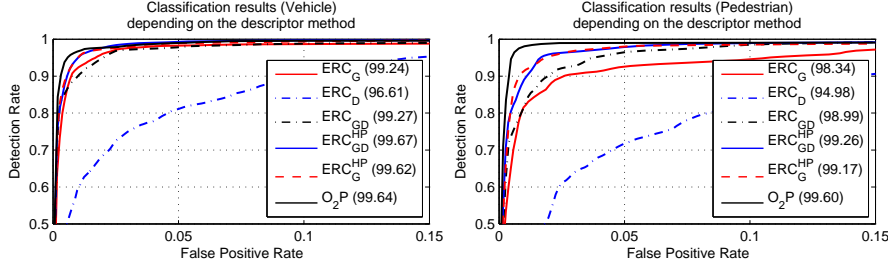


Figure 5.9: Comparison of the baseline region descriptor ERC_G on the *DUS* dataset with our proposed depth-based variant ERC_D and the O_2P descriptor (Carreira et al., 2012), demonstrated using the Vehicle and Pedestrian classes. Again, we additionally show the area under curve (AUC, in %) in the legend.

	Gnd	Veh	Ped	Sky	Bld	Avg	Dyn	#S
Individual Stixels	97.5	89.5	83.7	91.0	93.2	91.0	86.6	586
Stixel-based Regions	96.4	87.0	79.4	87.6	91.1	88.3	83.2	65
GPB-OWT-UCM ¹	93.1	72.5	74.5	89.8	87.6	83.5	73.5	142

¹Arbeláez et al. (2011)

Table 5.3: Impact of using Stixels and Stixel-based regions on the *DUS* dataset. We assign ground truth semantic labels, given by an oracle, via majority vote to Stixels and Stixel-based regions respectively and compare it again with the ground truth labeling image. The last column (#S) indicates the average number of segments per image.

To this end, we replace ground truth regions with inferred Stixels and our Stixel-based regions (*c.f.* Section 5.1.1), but assign the ground truth semantic label that is provided by an oracle instead of using a classifier. Assignment of the ground truth label is done via pixel-level majority vote. This experiment tells us how much scene labeling accuracy is lost by using Stixels as primitive element and how our Stixel-based region generation performs compares to other superpixel approaches. It also defines the upper bound that we can achieve with our Stixel-based scene labeling approach.

Table 5.3 shows the result of this experiment. While many small image segments allow to better reconstruct the ground truth segments and thus yield better results, fewer and larger segments are preferred for region classification, as they contain more usable image content and require less classifier executions. We compare individual Stixels and our Stixel-based regions against the GPB-OWT-UCM superpixel hierarchy of Arbeláez et al. (2011). When using individual Stixels, the maximum achievable average IOU score is 91%. Our Stixel-based proposal regions reduce this upper bound further to 88.3%, but reduce the number of segments by factor 9. In contrast, choosing an interme-

Regions		SLIC ¹ / Stixel-based (ours)				
Desc.	O ₂ P ²	ERC _G	ERC _D	ERC _{GD}	ERC _G ^{HP}	ERC _{GD} ^{HP}
Gnd	42.9/–	76.1/82.4	77.3/82.8	82.0/82.8	82.1/82.8	81.8/82.8
Veh	21.6/–	37.9/64.8	29.2/50.8	36.5/61.8	48.4/63.9	50.4/62.2
Ped	25.3/–	29.7/41.9	23.3/42.9	33.6/51.9	39.9/53.6	42.8/53.9
Sky	52.9/–	51.1/30.9	17.7/ 6.8	21.5/30.4	53.7/29.0	32.8/28.3
Bld	35.6/–	47.8/51.0	47.2/52.1	47.2/53.0	53.4/53.8	52.8/53.4
<i>Avg</i>	35.7/–	48.5/54.2	38.9/47.1	44.2/56.0	55.5/56.6	52.1/56.1
<i>Dyn</i>	23.5/–	33.8/53.3	26.3/46.8	35.1/56.8	44.2/58.8	46.6/58.1

¹Achanta et al. (2012) ²Carreira et al. (2012)

Table 5.4: Impact of different ERC descriptor variants and region proposals on segmentation accuracy. We compare our Stixel-based regions against SLIC segments and the ERC region descriptor against O₂P. Copied from (Scharwächter et al., 2013).

diate level in the GPB-OWT-UCM hierarchy yields lower ground truth reproduction capability despite more image segments. This is a clear indicator that Stixels and in particular our Stixel-based regions are well-suited to serve as primitive element for region-level scene labeling.

In the following, we perform fully automated scene labeling to evaluate the joint region generation and classification process, without any oracles involved. For region generation, we use $\Delta z_{\max} = 2\ m$, resulting in an average of 44 regions per image. As grayscale-based reference for region generation we use SLIC superpixels, parametrized to obtain approximately 150 regions per image. SLIC has been compared to various state-of-the-art superpixel methods and shows best boundary adherence with the least computational cost (Achanta et al., 2012). Table 5.4 shows the results for all combinations of region generation (SLIC vs our Stixel-based) and descriptors (O₂P vs our ERC variants).

It is interesting to see that in almost all cases, we yield better results when using our Stixel-based regions instead of SLIC superpixels, which is consistent with the results in Table 5.3. Another insight is that ERC_G notably outperforms O₂P on SLIC segments even though O₂P was superior in our ROC evaluation on ground truth segments, *c.f.* Figure 5.9. This result might be explained by the often reported robustness of coding-based descriptors against large amounts of clutter and occlusion (Grauman and Darrell, 2005; Moosmann et al., 2008). Note that O₂P does not involve any descriptor coding at all (Carreira et al., 2012). We furthermore highlight that we treat O₂P as an

	Gnd	Veh	Ped	Sky	Bld	Avg	Dyn	RT
M/SIFT-ERC _G ^{HP} ¹	87.5	66.2	53.4	51.4	61.1	63.9	59.8	544
M/SIFT-ERC _G ^{HP} +reg	87.6	68.9	59.0	57.6	60.2	66.7	64.0	571
S/Haar-ERC _G ^{HP}	87.6	61.8	51.5	55.2	60.9	63.4	56.7	23
S/Haar-ERC _G ^{HP} +reg	87.6	67.4	57.8	61.4	60.1	66.9	62.6	50

¹these numbers corresponds to the ERC_G^{HP} result in Table 5.4, re-computed with our extended Stixels.

Table 5.5: Improvements based on our proposed spatio-temporal regularization on the *DUS* dataset. We show use the ERC_G^{HP} region-encoding method with multi-scale SIFT and single-scale Haar descriptors. The last column shows the runtime (RT) of the approach in milliseconds, not taking into account stereo depth map and Stixel computation. Copied from (Scharwächter et al., 2014).

appearance-only baseline and hence did not evaluate it on our Stixel-based regions.

Table 5.4 is reproduced from the original publication in (Scharwächter et al., 2013). Note that at the time of that publication, we did not use the extended Stixel model that we introduced in Section 4.3. Instead, we used the Stixel algorithm from Pfeiffer and Franke (2011a). In Table 5.5, the best performing combination of Table 5.4 (ERC_G^{HP}) is re-computed with our extended Stixels as primitive element, to serve as baseline for further experiments. This change notably improves the overall performance, from 56.6% average *IOU* to 63.9%, which clearly supports our extended Stixel model and the statement that good initial image regions are crucial for the overall performance of the pipeline.

5.3.3 Spatio-temporal Regularization

So far, the presented scene labeling results are inferred on a frame by frame basis, without any spatial or temporal regularization involved. In this section, we incorporate and evaluate our smoothing approach from Section 5.2. Table 5.5 shows *IOU* performance for two variants of our approach, first without and then with regularization applied (“+reg”). The positive effect of regularization is quite apparent. We show results with both multi-scale SIFT and single-scale Haar descriptors because Haar descriptors deliver a very good trade-off between accuracy and runtime, as indicated by the last column (“RT”). We will discuss this aspect in more detail later in Section 5.3.4. One interesting aspect is that despite the simpler “S/Haar” features, global average performance after regularization is in fact slightly higher compared to using “M/SIFT” features (66.9% vs 66.7%), which shows that regu-

larization can compensate for the performance drop caused by using simpler features. We will refer to the regularized version of our algorithm as *Stixmantics* in the following. Table 5.5 is a summary of the results originally published in (Scharwächter et al., 2014), where “M/SIFT-ERC_G^{HP} +reg” corresponds to *Stixmantics* and “S/Haar-ERC_G^{HP} +reg” to *Stixmantics* (real-time).

*Important note
about the evaluation
metric used in this
section. Please read.*

Note that in contrast to all other results in this dissertation, the IOU results in Table 5.4 and Table 5.5 are computed with a slightly different evaluation metric: As stated in the last paragraph of Section 3.2.2, pixels that are not annotated should not contribute to the result. However, in those two tables they are taken into account in terms of false positives, which explains the generally lower scores. We adapted the evaluation metric after publication of (Scharwächter et al., 2013) and (Scharwächter et al., 2014) to make it consistent with other publications. When we compare our *Stixmantics* results later in Section 5.3.5 (Table 5.6), we again follow the established evaluation protocol from Section 3.2.2.

One general problem with the IOU measure is its pixel-wise nature. It is strongly biased towards close objects that take up large portions of the image and small objects of the same class barely contribute to the final score. The same holds true for small temporal fluctuations in the result. To account for this fact, we provide an additional pseudo object-centric evaluation that better captures the effect of our proposed spatio-temporal regularization. Figure 5.10 shows the number of detected objects o_i over the frames i , for the baseline approach (left) and the regularized approach (right). To approximate the number of objects, we count each closed image region with identical class label as one instance. Although the absolute number of objects is non-informative without ground truth data, a lower temporal fluctuation in the number of objects indicates stronger temporal consistency. We define a temporal fluctuation measure TF_i at frame i as sliding mean squared deviation to the sliding average \bar{o}_j :

$$TF_i = \frac{1}{2w+1} \sum_{j=i-w}^{i+w} (o_j - \bar{o}_j)^2 \quad \text{with } \bar{o}_j = \frac{1}{2w+1} \sum_{k=j-w}^{j+w} o_k, \quad (5.14)$$

where w is the temporal window size and is set to 10 frames in our evaluation. We show the TF_i score as $\bar{o}_i \pm TF_i$ for each frame i as solid band in the background of Figure 5.10. Our regularized model consistently outperforms the baseline throughout all object classes w.r.t. the TF measure. Some qualitative results of our region-level classification approach are shown in Figure 5.11.

5.3.4 Runtime

The region-level scene labeling approach in this chapter is implemented single-threaded in C++ and executed with an Intel i7-3.33

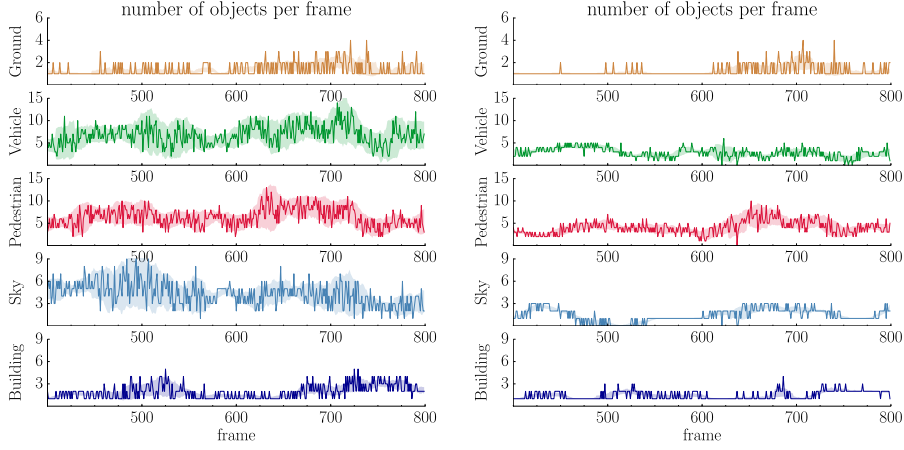


Figure 5.10: Number of detected objects per frame, for the baseline (left) and our regularized approach (right). We show an excerpt from frame 400 – 800 of the test_2 sequence. The TF_i score is shown as solid band in the background, illustrating the strength of local temporal fluctuations in the labeling decision. Our regularized model clearly provides stronger temporal consistency.

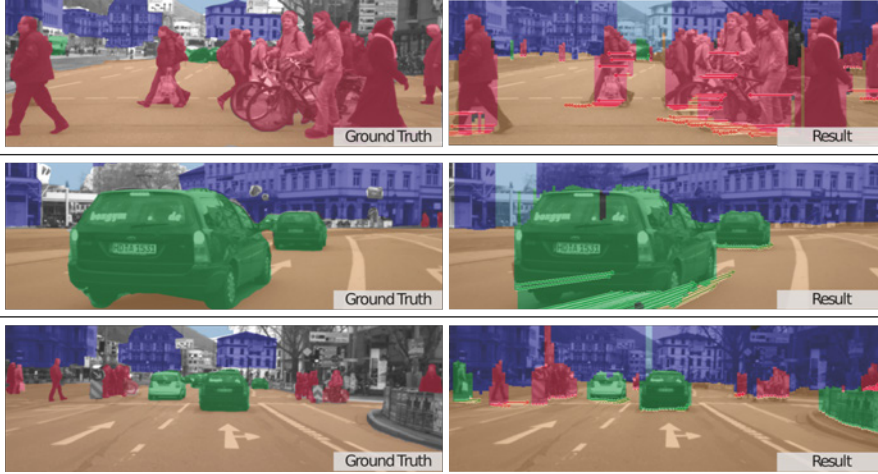


Figure 5.11: Comparison of results obtained with our region-label scene labeling model against ground truth labels. Colors denote the recovered semantic object classes. Arrows at the bottom of each underlying Stixel depict the ego-motion corrected 3D velocities of other traffic participants.

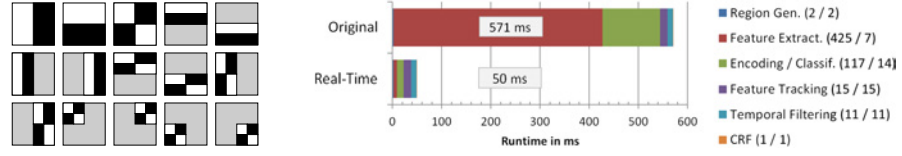


Figure 5.12: 8×8 pixel Haar wavelet basis feature set (left). White, black and gray areas denote weights of +1, -1 and 0, respectively. Adapted from (Stollnitz et al., 1995). Average runtime in milliseconds of the proposed Stixmantics approach and the real-time variant (right). This plot only compares the runtime of region classification. Stereo depth map computation (50 ms) and extended Stixel computation (30 ms) are not taken into account (see text).

GHz CPU and an NVIDIA GeForce GTX 770 GPU. The GPU is used for pixel-level classification that serves as input for the extended Stixel model, as well as KLT feature tracking to enable the track-level temporal regularization.

Figure 5.12 (right) shows that SIFT feature extraction (red) and random forest encoding coupled with SVM classification (green) are the computational bottlenecks of the system. We therefore implemented a variant with much faster inference time, by replacing the multi-scale SIFT features with simpler and faster descriptors. In particular, we use single-scale 8×8 pixel features derived from a 2D Haar wavelet basis resulting in a 15-dimensional descriptor (Stollnitz et al., 1995), see Figure 5.12 (left). The results in Table 5.2 and Table 5.5 show that the quantitative drop in accuracy is only minor. Additionally, a weaker random forest encoder is applied using 5 trees with 50 leaves (instead of 500 leaves) each. Encoding is faster due to shallower trees and the shorter histograms also result in faster SVM classification. In doing so, the computation time is greatly reduced by an order of magnitude to 50 ms per image on average, see Figure 5.12 (right). In Table 5.6 we print the overall runtime, including the time needed to compute stereo depth maps (50 ms) and extended Stixels (30 ms). Note that when following a pipelining strategy, we are able to operate at a real-time frame rate of 20 Hz.

5.3.5 Comparison to Reference Methods

In this last experiment section, we compare our Stixmantics region-level scene labeling approach with the reference methods introduced in Section 2.4.3. Table 5.6 shows the qualitative performance and runtime of all reference methods and our approach. In terms of accuracy, our approach is in the middle field of all reference methods. In terms of runtime, it is among the fastest methods and in contrast to all other approaches, our modular design allows for further runtime improvements by means of component pipelining. The best performing method (Layered Interpretation, Liu et al. (2015)) is still slightly

faster and also more accurate than our result, but their approach is not modular and can thus not achieve higher refresh rates by pipelining components. These findings are reminiscent of the coarse labeling results in the previous chapter (*c.f.* Table 4.7 in Section 4.4.6), except that now more fine-grained classes can be differentiated. It is also interesting to note that in the coarse label setup, all reference methods have been very close in performance. Moving to the finer label setup, performance differences become more apparent.

	Gnd	Veh	Ped	Sky	Bld	Avg	Dyn	RT
STF first level ¹	79.8	39.2	25.8	78.1	44.6	53.5	32.5	–
STF full ¹	93.6	70.6	46.0	66.6	73.4	70.0	58.3	125 ms ⁷
Darwin unary ²	94.5	64.4	26.7	90.6	84.9	72.2	45.5	250 ms
Darwin pairwise ²	95.7	68.7	21.2	94.2	87.6	73.5	44.9	1.20 s
ALE ³	94.9	76.0	73.1	95.5	90.6	86.0	74.5	111 s
PN-RCPN ⁴	96.7	79.4	68.4	91.4	86.3	84.5	73.8	2.8 s
Layered Int. ⁵	96.4	83.3	71.1	89.5	91.2	86.3	77.2	110 ms
Stixmantics (real-time) ⁶	93.8	78.8	66.0	75.4	89.2	80.6	72.4	130 ms

¹Shotton et al. (2008) ²Gould (2012) ³Ladický et al. (2010)

⁴Sharma et al. (2015) ⁵Liu et al. (2015)

⁶corresponds to “S/Haar-ERC_G^{HP} +reg” in Table 5.5, with updated evaluation metric to match the remaining numbers.

⁷measured runtime with available code was 750 ms, but we print faster timings as reported in Shotton et al. (2008).

Table 5.6: Quantitative results on the *DUS* dataset compared to reference methods from Section 2.4.3. Last column reports runtime (RT), as reported earlier in Table 4.9. We use our Stixmantics (real-time) variant for comparison, as this provides the best trade-off in terms of accuracy and runtime.

5.4 DISCUSSION

In this chapter, we introduced a scene labeling approach that uses Stixels as primitive scene elements by grouping them into larger image regions based on depth proximity and subsequently classifying these regions by means of a Bag-of-Features (BOF) approach. We found several interesting insights when comparing this approach to the pixel-level classification scheme from Chapter 4. Most importantly, the region-level approach yields higher precision for fine grained object classes, while the pixel-level approach can better predict the coarse semantic layout of the scene, *c.f.* Table 5.2. We attribute this finding to two independent reasons: (1) For coarse semantic classes

in outdoor traffic scenes, color, texture, and depth are more homogeneous, *i.e.* the ground plane has a more or less constant depth gradient and vegetation and sky have a mostly constant color. Minor changes to this constant model can be well captured by inspecting a small local window around each pixel. On the other hand, for fine-grained classes such as vehicles, pedestrians, and buildings, texture plays a more significant role, which is better captured with BOF histograms on the region level. (2) Another reason we see is the number of available training samples, which is fairly limited on region level, in particular for coarse semantic classes such as ground and sky, with mostly one instance per annotated image compared to thousands of samples per image on the pixel level. This limits the generalization performance of the region-level classifier. As this limited generalization also translates into sporadic mis-classifications of regions over time, we extended the region-level approach with a spatio-temporal regularization concept and showed that it improves labeling accuracy.

Having two approaches on the pixel level and the region level with their individual benefits and drawbacks now raises the question how to combine them to a solution that leverages the benefits of both models. A straight-forward combination would be to execute them consecutively in the following way: predict coarse semantic scores (*i.e.* ground, sky, vegetation, and obstacle) on pixel level to extend the Stixel model, then leave all Stixels with a label ground, sky, or vegetation untouched and apply the region-level approach only to Stixels with obstacle label to further subdivide obstacles into more fine-grained labels such as pedestrian, vehicle, and building. While we indeed propose this approach of combination, *i.e.* consecutive in terms of execution and hierarchical in terms of label granularity, following this idea with exactly the method proposed in Chapter 5 would result in a solution that is unsatisfactory from a conceptional point-of-view. This is because both methods are completely unrelated w.r.t. feature descriptors and classifier models used to realize them. Consider image regions covering obstacles in the scene. First, the pixel-level classifier extracts a set of local feature channels and classifies all pixels with model A. After generating Stixels and grouping them to proposal regions, new features (*e.g.* SIFT or Haar) are extracted, vector quantized with model B and added to a histogram that is finally classified with model C. This double effort of feature extraction, classification, and vector quantization should ideally be avoided or reduced to a minimum. To solve the aforementioned shortcomings of a straight-forward combination of the two approaches, the following chapter proposes a more coherent way of bringing both methods together while maximizing the re-use of computational resources.

Part III

COMBINING PIXEL AND REGION LEVEL

A JOINT MODEL FOR EFFICIENT ENCODING AND CLASSIFICATION

CONTENTS

6.1	Encode-and-Classify Trees	114
6.1.1	Conflicting Demands of Encoding and Classification	116
6.1.2	Sub-trees for Feature Encoding	117
6.1.3	Range Restriction	118
6.1.4	Training details	118
6.2	Final Concept	120
6.3	Experiments	120
6.3.1	Conflicting Parameters	120
6.3.2	Comparison to Reference Methods	122
6.3.3	Runtime	124

Chapter 4 and Chapter 5 introduced two approaches to the scene labeling problem on different levels, where both had their individual benefits and drawbacks. In particular, the pixel-level [RDF](#) classification approach is better at inferring the coarse semantic layout of outdoor traffic scenes, while the region-level [RDF](#) histogram encoding approach is better in differentiating fine-grained object classes. In this chapter, we propose a concept that combines the benefits of both approaches by integrating them into a single [RDF](#) model. A joint model is not only appealing from a conceptual point-of-view, it also avoids computational overhead, which is an important factor for application in a vehicle.

We will find that it is challenging to combine pixel-level classification and region-level encoding, due to conflicting demands in terms of parameters which govern the performance of both approaches. To overcome these challenges, we introduce a dedicated sub-tree structure for feature encoding, together with a special way of training the trees. The resulting model provides a good balance between classification and encoding performance while maximizing re-use of computational resources. We will then discuss the final concept we put forward in this dissertation for efficient automotive scene labeling. Parts of the work presented in this chapter have also been published in ([Cordts et al., 2017a](#)).

6.1 ENCODE-AND-CLASSIFY TREES¹

The concept we present in this chapter aims at unifying pixel-level classification that we use for our extended Stixel model (*c.f.* Chapter 4) and encoding of local image features that we use for region-level histograms (*c.f.* Chapter 5) into a single random forest model. In this way only a single inference pass is required on the pixel level, so that the extra cost of feature extraction and quantization introduced in Section 5.1.2 can be avoided. We accordingly call our concept *encode-and-classify trees*. The main idea of our concept is that the decision forest used for pixel-level classification in Section 4.2 already provides an explicit separation of the underlying feature space that can be used for region encoding exactly as described in Section 5.1.2. The only difference is that the type of input feature descriptor, *i.e.* SIFT or Haar, is replaced with the dense multi-cue feature channels evaluated earlier in Section 4.2.1. In doing so, the trees not only infer pixel-level semantic class posteriors, but also extract a set of multi-cue semantic textons maps, one for each tree, where the textons are learned from data and are optimized for the specific set of target classes.

Our method is in fact highly related to the semantic texton idea proposed by Shotton et al. (2008), which has been discussed earlier in Section 2.4.3. We will therefore briefly review their approach again and point out similarities and differences to our approach in terms of concept and use case. The semantic texton idea consists of two parts: The first part is the Semantic Texton Forest (STF), which extracts texton maps from small local image patches by encoding each pixel with the ID of the leaf node the pixel falls into. Additionally, its leaf node class posterior distribution provides a rough semantic prior, *c.f.* Figure 2.10. The second part is the Bag-of-Semantic-Textons (BOST) histogram, which captures the distribution of textons over an image region, where the histograms of all trees in the STF are concatenated and further extended by attaching the average class class posterior distribution from the STF leaf nodes as a region-level semantic prior, *c.f.* Figure 2.11. The final labeling is then obtain by using this BOST descriptor in a modified version of the TextonBoost approach (Shotton et al., 2006). For more information, please refer to Section 2.4.3 or (Shotton et al., 2008) directly.

On a higher level, the two parts in Shotton et al. (2008), *i.e.* STFs and BOST descriptors, are conceptually similar to the approach followed in this dissertation. In both cases, an RDF is used to predict pixel-level semantic labels and texton maps. Histograms of textons are then accumulated over regions to refine these initial labels. Furthermore, both methods rely on supervised learning to extract discriminative textons.

¹ Parts of this section have previously been published in (Cordts et al., 2017a), where the encode-and-classify concept is original work of Timo Rehfeld (myself). Those parts have been copied verbatim from the original publication.

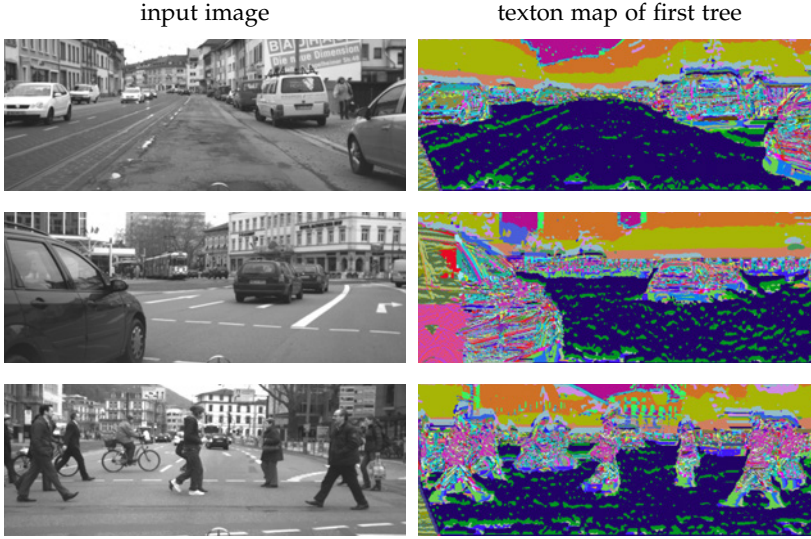


Figure 6.1: Texton map examples on the *DUS* dataset, showing the color-coded leaf node IDs of the first E&C tree. Every color corresponds to a unique leaf node. Regions that are homogeneous in any of the feature channels we use, *i.e.* color or depth gradient, also typically fall into the same texton bin.

However, there are several important differences in terms of how the two concepts are applied and combined. The most important difference is the target use case of the *STF*. In (Shotton et al., 2008), the main purpose of the *STF* is to extract textons for the *BOST* descriptor, which is then used in a second *RDF* to infer the final pixel-level labeling. The semantic results of the *STF* are rather a weak pixel-level prior, compared to the final result. In contrast, our goal is to combine texton extraction and labeling into one model, where labeling accuracy needs to be optimized to support our extended Stixel model and texton extraction needs to be optimized for our Stixel-based region classification. It turns out that this is difficult to achieve in one model because the two objectives have conflicting demands on the tree structure and training parameters. We will discuss this in more detail in Section 6.1.1. Another important difference is that we compute multiple low-level feature channels based on color, texture and depth, yielding multi-cue textons, whereas the textons in (Shotton et al., 2008) are learned from color images only.

In Figure 6.1 and Figure 6.2 we show examples of texton maps extracted with our E&C trees on images of the *DUS* and *KITTI* dataset, where every color indicates a different texton. It is interesting to see how the spatial structure of these maps visualizes the class-dependent complexity of the classification problem that we have also seen in previous experiments: classes such as ground and sky are covered by only few different textons, which indicates that a local pixel-level decision is seemingly sufficient. At the same time, vehicles and pedes-

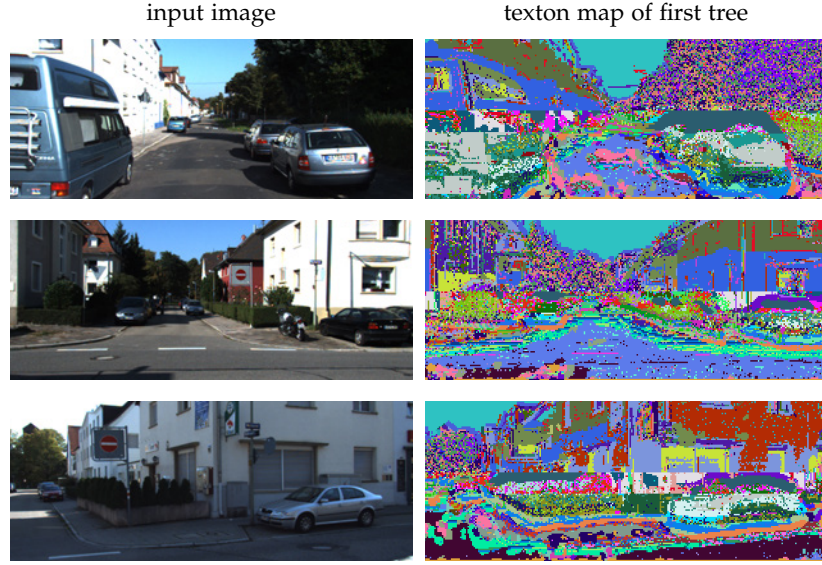


Figure 6.2: Texton map examples on the *KITTI* dataset, showing the color-coded leaf node IDs of the first E&C tree. Every color corresponds to a unique leaf node. Regions that are homogeneous in any of the feature channels we use, *i.e.* color or depth gradient, also typically fall into the same texton bin.

trians are covered by a many different textons. This is a result of more fine-grained split tests in the decision tree due to the higher complexity of the problem. We can also see a dataset-dependent effect: In images of the *DUS* dataset, the road surface is represented homogeneously with only a few textons, which is not the case for the *KITTI* dataset. This is because the depth information of the *KITTI* dataset is more noisy, so that the textons focus more on color, rather than depth.

6.1.1 Conflicting Demands of Encoding and Classification

The encode-and-classify concept put forward in this chapter is based on the observation that pixel-level classification and discriminative texton generation have different demands on the algorithm parameters. In particular, we find that good pixel-level accuracy requires medium to large patches while texton histograms are more discriminative if textons are extracted on rather small patches. This finding is consistent with our previous experiments in Section 4.4 (Figure 4.9) and Section 5.3 (Figure 5.8) and is evaluated explicitly again in Section 6.3.1. Our intuition is that visited node IDs of neighboring pixels are less correlated when smaller patches are used, so that in turn the benefit of histogram pooling is more pronounced. Furthermore, we observed earlier that deeply trained trees improve pixel-level accuracy. However, with increased tree depth the number of nodes, *i.e.* the histogram length and with it the region classification runtime in-

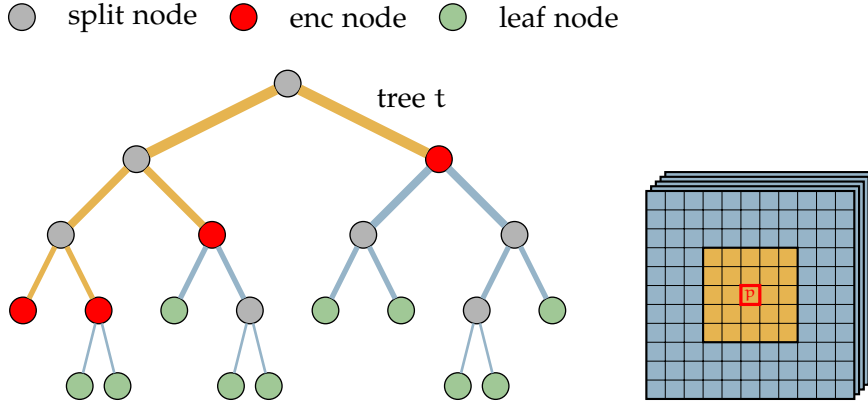


Figure 6.3: The proposed encode-and-classify tree structure for pixel-level classification and texton extraction. Encoder nodes define a sub-tree that operates on small local patches of size $S_1 \times S_1$ to obtain highly discriminative texton histograms. The remaining nodes have full access to all pixels in the larger region $S_2 \times S_2$ for accurate classification.

creases quickly. This fact has been pointed out in the conclusion of [Shotton et al. \(2008\)](#) and is also validated experimentally again in Section 6.3.1. To overcome these conflicting demands on the tree parameters, one could naturally use separate models for pixel labeling and region encoding, similar to [Shotton et al. \(2008\)](#), which however comes at the cost of additional runtime. With our encode-and-classify trees, we aim at combining the ideal operating points for both individual tasks within a single dual-use model, by introducing two concepts: sub-trees for feature encoding and range restriction, which are presented in Section 6.1.2 and Section 6.1.3 respectively. By combining both concepts, we decouple encoding and classification to a great extent: we gain full control over the histogram length to find the best trade-off between region-level accuracy and runtime and at the same time keep pixel-level labeling accuracy high. Training details are given in Section 6.1.4.

6.1.2 Sub-trees for Feature Encoding

The first problem is to decouple tree depth for encoding and classification, which requires shallow and deep trees respectively. To solve this problem, we propose to use a sub-tree dedicated for feature encoding. In contrast to using leaf nodes of the entire decision tree ([Moosmann et al., 2008](#)) or using all nodes from multiple levels in the tree ([Shotton et al., 2008](#)) to identify textons, we use explicit *encoder nodes*, which are special nodes in the tree, as depicted in Figure 6.3. Starting from the root node, encoder nodes can be seen as leaf nodes of a sub-tree that have the main purpose of texton extraction. At the same time, every encoder node can be the root of a following sub-tree dedicated for

pixel labeling, so that there is always exactly one encoder node on the path from the root node to any leaf node. Only those explicit encoder nodes contribute to the texton histogram. During inference, the tree is traversed as before from root to leaf to obtain the pixel-level classification decision. During traversal, passing an encoder node with a unique ID that identifies the texton is guaranteed, so that a texton map can be generated on-the-fly without additional runtime. Figure 6.3 visualizes the encoding and classification sub-trees with yellow and blue edges respectively.

6.1.3 Range Restriction

The second problem is to decouple patch size for encoding and classification, which requires small and large patches respectively. We propose to solve this problem by restricting the range of the encoding sub-tree: split tests of the texton extraction sub-tree are restricted to be within a small local range, while the following sub-tree has access to all pixels within a larger range, as indicated by the corresponding colors of tree edges and patches in Figure 6.3.

To obtain such a tree structure, training of the encode-and-classify trees is performed in two steps. First, we generate unary pixel tests within the small $S_1 \times S_1$ patch around a pixel of interest. We train all trees to full depth and then prune them back bottom up to the desired number of encoder nodes, which gives us full control over the histogram length. In each pruning iteration, we randomly select a node from the list of all candidates for pruning, where a candidate is a split node with both children being leaves, and prune its children, so that the split node becomes a new leaf. Second, we start at the encoder nodes and continue training in the same way, but with access to the larger $S_2 \times S_2$ patches to improve the pixel-level labeling performance.

6.1.4 Training details

As discussed, the main reason our encode-and-classify trees need to predict good pixel-level semantic label scores is their use for our extended Stixel model. In Section 4.4.4, we found that inferring the full set of all categories directly on pixel level does not yield sufficiently high accuracy, but the model works well to infer the coarse layout of the scene (ground, object, vegetation, and sky), so that we decided to only use those coarse labels for the Stixel extension. Consequently, we would only need to train the model with the objective to separate those coarse classes. However, previous experiments showed that it is beneficial to still use the full set of object classes during training, because it improves the discriminativity of the texton maps used later for region-level encoding. Therefore, we train with the full set

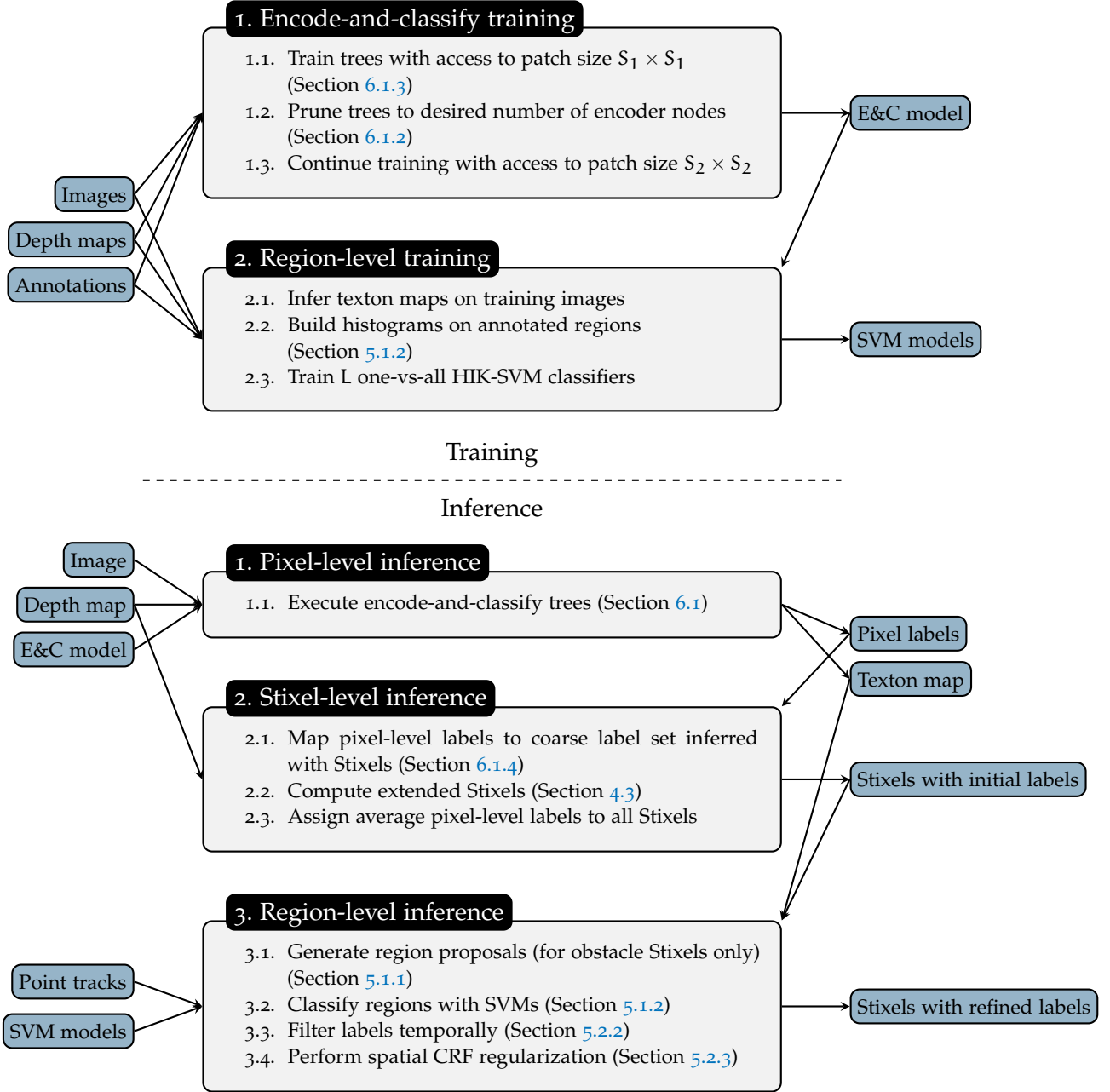


Figure 6.4: Training and inference overview of the final scene labeling concept proposed in this dissertation. Small boxes (left and right) reflect the inputs and outputs of the logical components (center). Arrows indicate the data flow, *i.e.* the dependencies between components. The logical components are broken down into the individual steps involved to generate the desired output. References to the respective sections are provided for further details.

of classes as objective and then combine classes to the coarse set after inference, following the same mapping introduced in Section 4.4.4.

6.2 FINAL CONCEPT

Given the encode-and-classify idea, which provides a link between the approaches discussed earlier in this dissertation, we are now able to propose the final concept we put forward for efficient scene labeling of outdoor urban street scenes. The concept involves all major components presented earlier in this work and integrates them in a way that is optimized for the particular use case of urban scenarios.

As already discussed in Section 5.4, we propose to combine the benefits of the pixel-level and region-level approach in a way that is consecutive in terms of execution and hierarchical in terms of label granularity. Inference starts with executing encode-and-classify trees for every pixel on a regular grid, where the grid resolution corresponds to the desired target resolution of Stixels. The resulting pixel-level labels are then used to compute the extended Stixel model. As discussed in Section 6.1.4, we map the inferred labels to the coarse label set that is used in Stixels prior to passing it to the Stixel algorithm. Based on our earlier findings, the region-level refinement is then only applied to all Stixels with class Obstacle. All other Stixels keep their label as inferred by the E&C trees and extended Stixels. Neighboring obstacle Stixels are grouped to region proposals and each region is classified based on the texon maps provided by our E&C trees, yielding a refined class label for every region. This second labeling decision is then additionally regularized in space and time (*c.f.* Section 5.2) and finally assigned back to Stixels. An overview of the training and inference steps is provided in Figure 6.4.

6.3 EXPERIMENTS

In this final experiment section, we will first cover the conflicting parameter requirements for encoding and classification as discussed in Section 6.1.1. Our joint encode-and-classify concept is further compared to using separate models for the two tasks, to support the effectiveness of the model. We finish with a comparison of our final scene labeling concept to the reference methods discussed in Section 2.4.3.

6.3.1 *Conflicting Parameters*

In Figure 6.5, we demonstrate the different demands on patch size for pixel classification and region encoding, by training separate standard RDF models for the two tasks. To assess region encoding accuracy, we generate and classify histograms on the segment level. As segments we use ground truth regions in the dataset, as well as seg-

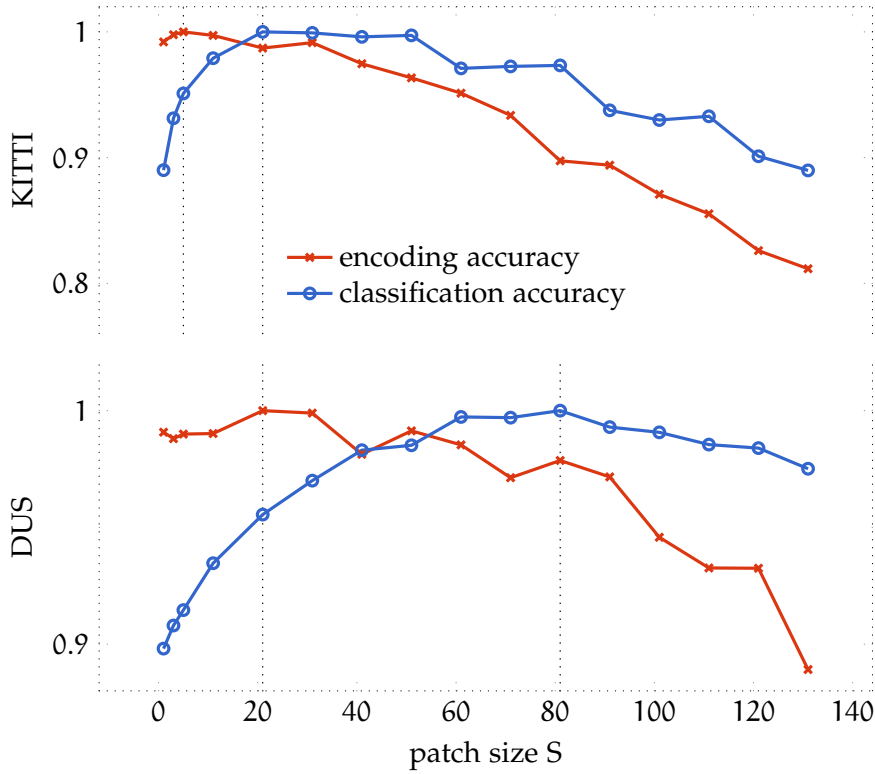


Figure 6.5: Encoding accuracy (red) and pixel classification accuracy (blue) as a function of patch size. Note that each curve is normalized such that its maximum value is 1. We observe that smaller patches are better for encoding, while medium to large patches are good for pixel classification. This trend is consistent on both datasets: *DUS* and *KITTI*. The maximizing patch sizes are highlighted for each curve.

ments obtained by greedily grouping superpixels using 3D proximity. Performance is reported using the average F-score over all classes. We normalized both curves to a maximum value of 1, as only the trends are important here. It can be seen that the highlighted maxima of both curves do not coincide, hence the maxima cannot both be obtained at the same time using a single standard RDF. Executing two separate models with different patch sizes would overcome this problem, however at the cost of additional runtime. In contrast, a single forest of our encode-and-classify trees with the restricted patch size $S_1 < S_2$ yields a performance close to both maxima, while having the same runtime as a single standard RDF as shown in Table 6.1. Despite the small remaining gap in accuracy compared to using two separate forests, we did not observe a drop of performance in the overall system, while affording an increased efficiency.

In Figure 6.6, we further investigate the trade-off between encoding accuracy and histogram length by plotting the region classification average F-score over all classes together with histogram length as a

Patch size S_1/S_2	standard RDF			E&C trees	
	21^1	81^1	$21/81^2$	$21/81$	$31/81$
encoding accuracy	1.0	0.979	1.0	1.0	0.999
classification accuracy	0.955	1.0	1.0	0.97	0.987
runtime [ms]	8	8	14	8	8

¹Single model with one patch size for both encoding and classification

²Two separate models with different patch sizes for encoding and classification

Table 6.1: Standard RDFs and encode-and-classify (E&C) trees compared on the *DUS* dataset with different patch size combinations. Accuracy is given as average F-score, relative to the maximum possible encoding and classification performance, according to Figure 6.5. Using our E&C trees with a smaller patch size S_1 for encoding and a larger one for classification S_2 yields a performance close to the individual maxima of standard RDFs. At the same time, they are more efficient compared to using two separate standard models for encoding and classification (center column).

function of tree depth D . We use the length as a proxy metric for runtime, as the runtime of our SVM classifier scales linearly with histogram dimensionality. Furthermore, convergence of the histogram length (*i.e.* number of leaf nodes) also indicates saturation of pixel-level classification accuracy, as the trees have eventually solved the training set and no new leaves need to be created. It is interesting to see that on both datasets the region-level classification accuracy saturates rather quickly, while the histogram length continues to grow, until it also starts to saturate, but much later. This finding supports our claim that high pixel-level classification accuracy requires deeply trained trees, while shallow trees with fewer leaf nodes are sufficient for textron extraction and also keep histogram classification runtime down. Our encode-and-classify concept combines both objectives in a single model, where only a sub-tree of an entire decision tree is used for textron extraction, and the number of encoder nodes can be chosen freely and independently of the remaining tree parameters.

For Figure 6.5 and Figure 6.6 we use 5 trees in total with 200 encoder nodes each, resulting in a 1000-dimensional region-level histogram (*c.f.* Figure 6.6). For the following experiments, we choose the patch sizes as $S_1 = 31$ and $S_2 = 81$ for *DUS* (*c.f.* Table 6.1) and $S_1 = 5$ and $S_2 = 21$ for *KITTI* (*c.f.* Figure 6.5).

6.3.2 Comparison to Reference Methods

For comparison with the reference methods of Section 2.4.3, we embed the E&C concept into our overall system as described in (Cordts

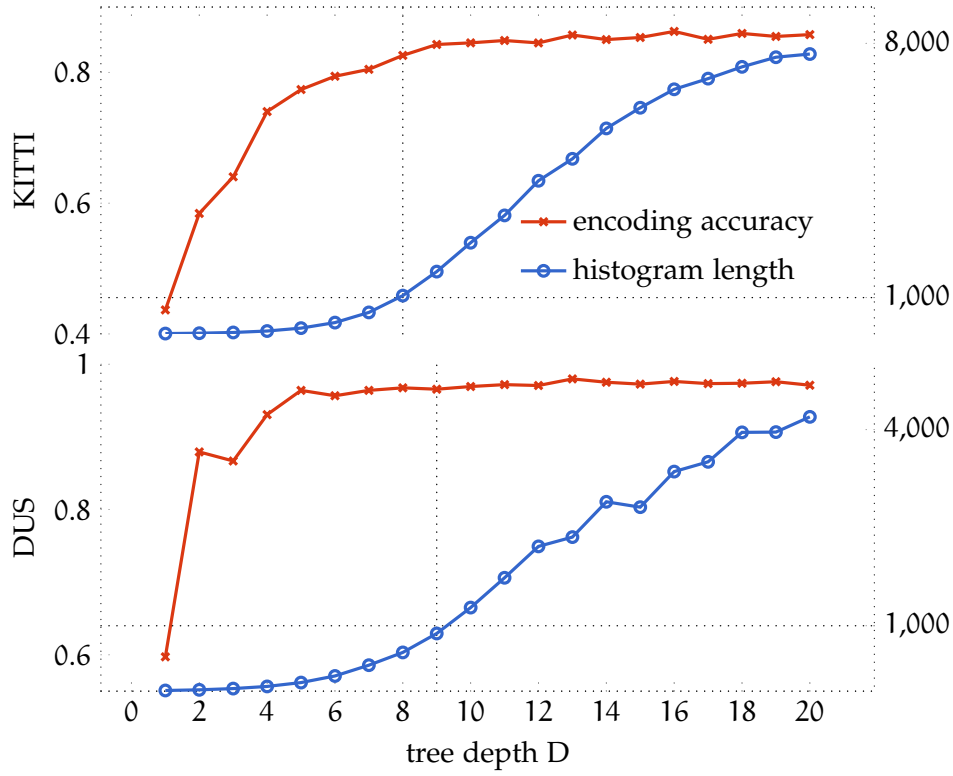


Figure 6.6: Encoding accuracy (red, left axis) and histogram length (blue, right axis) as a function of tree depth. It can be seen that with increased depth encoding accuracy saturates rather quickly, but the histogram length and thus histogram classification runtime grows continuously. This shows that limiting the number of encoder nodes, *i.e.* the histogram length, is important to set the optimal trade-off between accuracy and runtime. We set the histogram length to 1000, as highlighted.

et al., 2017a). The system in this paper is a joint development with Marius Cordts, where the region generation process from Section 5.1.1 is replaced with a segmentation tree and the CRF inference from Section 5.2 is replaced with a tree-structured CRF inference. In addition, we further employ individual object detectors for vehicles and pedestrians to improve performance of these two important classes of dynamic objects. These extensions are contributed by Marius Cordts and are not part of this dissertation. More details on the exact algorithm configuration are given in (Cordts et al., 2017a). Table 6.2 shows the results of this system (“Ours”) on the *DUS* dataset. We further show results of a variant without object detectors (“-DT”) that more closely reflects the concept proposed in Section 6.2. Compared to our Stix-mantics approach from the previous chapter, we increase accuracy and slightly reduce computation time (118 ms vs. 130 ms), thanks to the more efficient E&C tree concept.

Overall, our results on the *DUS* dataset are comparable to the best performing reference method (Layered Interpretation, Liu et al. (2015)) in terms of both accuracy and runtime. In the dynamic object score we surpass all reference methods. Also, our model consists of several building blocks that can be pipelined to achieve refresh rates of 20 FPS for both stereo depth maps and scene labeling results. This is not possible with the approach of Liu et al. (2015), which computes depth maps and semantics jointly. This is an important benefit of our approach for real-time in-vehicle application.

On the *KITTI* dataset, our model outperforms all reference methods in terms of accuracy. In particular, the scores for the two dynamic object classes (vehicles and pedestrians) are significantly higher, c.f. Table 6.3.

6.3.3 Runtime

Runtime is evaluated using an Intel Core i7 CPU and a NVIDIA GTX 770 GPU. We report the total amount of time spent to label a single image including the computation of all cues. For stereo depth maps, we assume 50 ms runtime. In Figure 6.7, we break down the overall system runtime of our method into individual component runtimes, to demonstrate the efficiency of the tree-structured models. The execution time of our E&C trees takes up less than 5% of the overall runtime, which clearly indicates its efficiency. We can further reduce the overall runtime in an online setup by pipelining the components. In doing so, we achieve a throughput of 20 FPS at the cost of one frame delay, as no individual component takes longer than 50 ms, and stereo depth maps, detectors, and point tracks can be computed in parallel.

	Gnd	Veh	Ped	Sky	Bld	Avg	Dyn	RT
STF first level ¹	79.8	39.2	25.8	78.1	44.6	53.5	32.5	–
STF full ¹	93.6	70.6	46.0	66.6	73.4	70.0	58.3	125 ms ⁷
Darwin unary ²	94.5	64.4	26.7	90.6	84.9	72.2	45.5	250 ms
Darwin pairwise ²	95.7	68.7	21.2	94.2	87.6	73.5	44.9	1.20 s
ALE ³	94.9	76.0	73.1	95.5	90.6	86.0	74.5	111 s
PN-RCPN ⁴	96.7	79.4	68.4	91.4	86.3	84.5	73.8	2.8 s
Layered Int. ⁵	96.4	83.3	71.1	89.5	91.2	86.3	77.2	110 ms
Stixmantics (real-time) ⁶	93.8	78.8	66.0	75.4	89.2	80.6	72.4	130 ms
Ours, -DT	96.2	82.9	68.9	83.3	89.6	84.2	75.9	118 ms
Ours, full	96.2	85.4	74.3	82.8	89.6	85.7	79.9	163 ms

¹Shotton et al. (2008) ²Gould (2012) ³Ladický et al. (2010)

⁴Sharma et al. (2015) ⁵Liu et al. (2015)

⁶our result from the previous chapter.

⁷measured runtime with available code was 750 ms, but we print faster timings as reported in Shotton et al. (2008).

Table 6.2: Quantitative results on the *DUS* dataset compared to reference methods from Section 2.4.3. Last column reports runtime (RT), as reported earlier in Table 4.9. These numbers corresponds to Table 2 in (Cordts et al., 2017a).

	Gnd	Veh	Ped	Sky	Bld	Veg	Avg	Dyn
STF full ¹	79.4	50.6	3.6	54.3	63.5	71.0	53.7	27.1
Darwin pairwise ²	89.8	75.8	8.0	81.6	83.4	85.8	70.7	41.9
ALE ³	89.9	76.0	24.8	81.2	84.9	86.6	73.9	50.4
Ours, full	89.7	79.1	51.2	77.5	79.6	77.7	75.8	65.2

¹Shotton et al. (2008) ²Gould (2012) ³Ladický et al. (2010)

Table 6.3: Quantitative results on the *KITTI* dataset compared to reference methods. These numbers corresponds to Table 3 in (Cordts et al., 2017a).

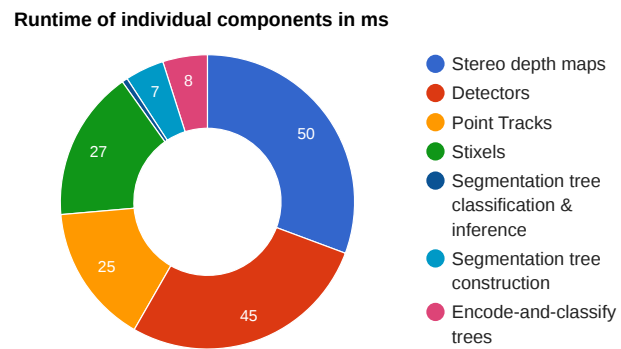


Figure 6.7: Breakdown of the 163 ms runtime reported in Table 6.2 into the individual components of our overall system. The E&C trees only use about 5% of the overall time, clearly indicating their efficiency.

DISCUSSION AND OUTLOOK

CONTENTS

7.1	Discussion and Limitations of the Approach . .	127
7.2	Future Perspectives	130
7.2.1	Segmentation Trees	131
7.2.2	Scene Labeling Datasets	131
7.2.3	Deep Learning	132
7.3	Final Thoughts	132

After introducing the final scene labeling concept of this dissertation in Chapter 6, we will now summarize the main contributions again, put them into perspective to other approaches and discuss limitations of the model. This discussion is complemented with an outlook towards potential improvements of the presented method and a brief discussion on how to use novel deep learning techniques, which are very promising in the field of scene labeling and machine learning in general.

7.1 DISCUSSION AND LIMITATIONS OF THE APPROACH

In the following, we will recapitulate and discuss our final approach with respect to the problem statement and requirements we listed in the beginning of this dissertation. This is important to analyze which goals were reached and which problems are potentially still open and need to be addressed further. This work provided several contributions to different aspects of automotive scene labeling, all of which have been listed earlier in Section 1.3. While all of these contributions are logical steps on the way to our final proposed concept, they are at the same time independent of each other, so that they can also be applied individually in a different context. After the general discussion of the final approach, we therefore group in-depth discussions according to the list of contributions in Section 1.3.

Overall, the scene labeling approach presented in this dissertation fulfills all conditions we stated in the initial problem statement (Section 1.2), but there are certainly some limitations that still need to be addressed. In general, our scene labeling concept is competitive to the reference methods and at the same time meets the computational requirements to deliver scene labeling results with real-time refresh rates of a typical video camera. However, a look at the final

results in Table 6.2 actually shows that the best reference methods are quite similar in terms of accuracy and also overall runtime to label a single image. A central benefit of our approach is visible when moving from single image processing to live video processing, where our modular algorithm design allows to pipeline different components and thus yield much higher refresh rates. This gives a very notable runtime improvement in practice. On the other hand, the multiple intermediate decisions that need to be made during execution of all sub-components make it quite hard to systematically minimize a global loss function during training of the algorithm parameters. End-to-end approaches or methods such as (Liu et al., 2015) or (Ladický et al., 2010) that jointly optimize semantics and depth are easier to optimize in that regard.

In terms of scalability to many object classes, our concept works well for the number of classes used throughout this dissertation, *i.e.* five to seven, depending on the task and dataset. However, in the last two years or so, the number of object categories in scene labeling datasets has increased to 20+ or even 30+, *c.f.* Section 7.2.2. It is in fact questionable whether the presented approach is able to scale to these requirements from a conceptual point-of-view. The reason is two fold: having more categories automatically means that context becomes more important, but our approach does not take context into account. Furthermore, a very large number of categories also means that image regions of individual classes are becoming smaller. This has a negative impact on our region classification approach, which benefits from pooling information over larger image regions. From the computational resource perspective, scalability also involves sharing low-level features across multiple classes, as opposed to extracting class-specific descriptors. In this regard our final model is actually quite efficient, with our E&C trees focusing on re-use of the same features for multiple classes and multiple steps in the processing chain. However, novel deep CNN architectures for scene labeling also provide these properties and inherently take into account context, which our approach does not. We will briefly discuss novel deep learning techniques later in Section 7.2.3.

In Section 1.1.2 we discussed different ways to represent the visual environment and argued for a model that is neither too specific, *i.e.* bounding boxes for individual object instances, nor too generic, *i.e.* individual labeled pixels. Throughout this dissertation we worked with Stixels as primitive scene elements and argued that Stixels are a good intermediate representation for outdoor urban scenes, as they describe the environment in a compact way. This was one of the central criteria our scene labeling concept should fulfill, *c.f.* Section 1.2. Using Stixels also allows to easily integrate the presented scene labeling concept into existing software with minimal development over-

head, as discussed in Section 1.2.2. In the following, we discuss the specific contributions from Section 1.3 in more detail.

INTEGRATION OF APPEARANCE INTO THE STIXEL MODEL The original Stixel representation is inferred based on depth information only. In Section 4.4.4 we were able to successfully show that our extension with appearance information from a classifier significantly improves the segmentation accuracy of the model. However, the proposed Stixel extension does not scale well to a large number of object classes. With our pixel-level classifier from Chapter 4 this is not a problem, as only a small number of labels are taken into account, but the computational complexity of the algorithm grows quadratically with the number of classes. An alternative solution that scales linearly has been introduced by Lukas Schneider in (Cordts et al., 2017b).

STIXEL-BASED REGION GENERATION AND CLASSIFICATION In Section 5.3.2 we were able to show that by using Stixels as primitive scene elements, we can rapidly generate larger region proposals. Our proposals resemble the size and shape of actual objects in the scene better than typical state-of-the-art superpixels or image segmentation techniques, which only use texture or color information. This is a very strong indicator that depth is a valuable cue for proposal generation. On the other hand, our way of grouping Stixels to regions is only a first demonstration of what can be done. The algorithm proposed in Section 5.1.1 is greedy, only uses a single scalar value (depth distance) as grouping criterion, and only produces one segmentation result. Capturing all kinds object classes with just one segmentation is quite difficult and only using depth for segmentation can also fail, *e.g.* if pedestrians are too close to a wall to be segmented out, although they are easy to detect when taking into account color or texture. Improvements can thus be made by adding more grouping criteria and forming a segmentation hierarchy rather than having just one layer. We will discuss this in more detail in Section 7.2.1.

TEMPORALLY CONSISTENT SCENE LABELING We were able to show that the track-level label integration concept proposed in Section 5.2.2 is able to reduce the effect of sporadic mis-classifications of regions over time, using the mathematic framework of recursive Bayesian filtering. The concept can be applied on both sparse and dense point correspondences, but we chose to use sparse point tracks, as they are typically already available in an automotive computer vision software stack. This allows the re-use of intermediate results and thus sharing of resources. However, one caveat of this concept is that the tracker is completely independent of Stixels, *i.e.* there are always some Stixels without tracks, so the labeling information cannot be

filtered over time. Only a designated Stixel tracker or a dense correspondence field (*i.e.* dense optical flow) would solve this problem.

A JOINT MODEL FOR ENCODING AND CLASSIFICATION As discussed earlier in Section 2.4.1, the scene labeling literature can be roughly separated into pixel-centric and region-centric approaches. The encode-and-classify trees introduced in this dissertation unify those two methodologies into one model by providing a pixel-level classification result as well as textons for region-level feature encoding. In this way the model not only practically helps to reduce computational overhead, it also bridges the gap between two competing paradigms. Compared to the semantic texton forest approach of [Shotton et al. \(2008\)](#), which is most closely related to our work, we significantly outperform their method on both datasets we analyzed.

A NOVEL STEREO VISION DATASET Without a doubt, dataset generation and (based on that) extensive parameter evaluations and control experiments are an important part of this dissertation. They provided several interesting insights into scene labeling with an outdoor setting and allowed to reveal the benefits and drawbacks of different approaches, most notably the performance differences of pixel-level and region-level methods. On a higher level, this data-driven approach also had a major influence on the structure of this dissertation and the final concept itself: Every chapter proposed a solution to a problem that typically came to light after experiments with labeled data in the preceding chapter, so that the overall concept is rather a composition of individual solutions. While this might sound undesirable in theory, it is in fact important in practice in terms of modular design and is in line with the approach initially proposed in Section 1.2.3. This data-driven approach to problem solving was only possible with annotated data. Nevertheless, the *DUS* dataset introduced in Chapter 3 is still fairly limited w.r.t. to number of annotated object classes and images. We therefore give an outlook to alternatives that have been developed concurrently in Section 7.2.2.

7.2 FUTURE PERSPECTIVES

For some of the shortcomings of the proposed approach that have been discussed in the previous section, there is a straightforward way to address them. In this section, we briefly outline three such potential improvements and refer to existing work that is going into this direction.

7.2.1 Segmentation Trees

As mentioned earlier, our region generation process is limited because it only generates one segmentation result, which typically does not work well for all object categories at the same time. Region proposal trees or segmentation trees are a straightforward extension to solve this problem by not using a single threshold but applying multiple ones. If these multiple segmentations form a tree structure, *i.e.* two or more regions on one level are grouped to one region on the next level, efficient tree-structured inference algorithms can be applied to perform *e.g.* CRF-based regularization of labels within the tree that also encodes parent-child relations and not only neighborhood relations within one level, as we did in Section 5.2.3. Another problem with our regions is that the decision to cut is based on depth information only. A possible extension of such a handcrafted scalar threshold would be to train a classifier that learns to group Stixels to regions based on multiple cues extracted from color, texture, and depth. In Cordts et al. (2017a), M. Cordts introduces a grouping classifier that, amongst other cues, also leverages the encode-and-classify concept proposed in this dissertation. Using this grouping classifier, a segmentation tree is constructed and efficient tree-structured CRF inference is performed to regularize the labeling decision within all levels of the tree.

7.2.2 Scene Labeling Datasets

There is no question that the sheer size of a dataset, *i.e.* number of annotated images, is an important factor to test the generalization performance of a method. However, what we also found to be very important is that unlabeled image regions should be avoided as much as possible during annotation, as they leave room for false positives. Another problem of most currently existing datasets, including the DUS dataset introduced in Chapter 3, is that the entire dataset is recorded in very similar locations or under similar conditions, *e.g.* always in the same city. This leads to overfitting and again hinders generalization to new unseen environments. The Cityscapes dataset of Cordts et al. (2016) tackles exactly these shortcomings and hence provides a very interesting testbed for further research in the field of urban scene labeling.

Another very interesting concept is to use artificial data for training. In the last years, several researchers have invested into this strategy and released datasets such as “Playing for Data” (Richter et al., 2016), Virtual KITTI (Gaidon et al., 2016) or the SYNTHIA dataset (Ros et al., 2016), where the latter contains more than 200.000 images with highly precise pixel-level semantic annotations and depth maps. Another benefit of this approach is that illumination and weather con-

ditions can easily be changed for the same scene. Pre-training with such data and then fine-tuning with real data seems to be a highly promising approach to obtain better models.

7.2.3 *Deep Learning*

In the last years, deep neural networks have dramatically reshaped computer vision research. According to the benchmark website of the Cityscapes dataset¹, all of the best performing scene labeling approaches rely on deep neural networks. This is due to multiple reasons we are going to discuss briefly in the following.

The two steps of features encoding and classification, which are essential to many computer vision problems (including the scene labeling approach presented in this dissertation) were typically solved separately, requiring intermediate decisions to be made. One of the key benefits of deep learning, in particular of deep CNNs in computer vision, is that they combine feature encoding and classification into a single model that can be fully trained from data with one joint optimization. This also allows to learn low-level features that are optimized for a specific task, whereas classical feature descriptors such as SIFT and HOG are handcrafted and rather generic. In fact, our encode-and-classify concept also learns features for region-level encoding in a supervised way from data, but the objective function is still optimized for pixel-level classification, rather than for the actual problem of region-level classification. Deep CNNs also provide a very successful alternative in terms of combining short-range pixel-level precision and long-range context information into a single model. For scene labeling in particular, novel network structures with learned interpolation layers play an important role, *i.e.* the Fully Convolutional Network (FCN) approach presented by Long et al. (2015).

Arguably, most of the novel deep learning models for scene labeling are still fairly expensive in term of computational resources and thus only feasible with high-performance GPUs, but the improvements in terms of accuracy and the rapid progress in the fields make it one of the key technologies for many applications, including self-driving vehicles.

7.3 FINAL THOUGHTS

The contributions made in this dissertation provide conceptual ground work that is relevant and interesting when applying scene labeling algorithms in the automotive domain. Of course, exciting new developments, in particular in the field of deep learning, are constantly increasing the level of accuracy, but many of the general aspects discussed in this work, such as temporal coupling, representation form

¹ www.cityscapes-dataset.com

of the results, proper dataset generation, modularization and pipelining or combination of multiple cues remain important. It is a great time to work in this fast paced field and I am looking forward to seeing in-vehicle applications for self-driving cars that rely on scene labeling as a central component.

BIBLIOGRAPHY

- Alexey Abramov, Karl Pauwels, Jeremie Papon, Florentin Worgotter, and Babette Dellen. Real-Time Segmentation of Stereo Videos on a Portable System With a Mobile GPU. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(9), 2012. (Cited on page 25.)
- Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11), 2012. (Cited on pages xv, 84, and 104.)
- Pablo Arbeláez, Michael Maire, Charless C Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5), 2011. (Cited on pages xv, xvi, 84, and 103.)
- Pablo Arbeláez, Bharath Hariharan, Chunhui Gu, Saurabh Gupta, Lubomir Bourdev, and Jitendra Malik. Semantic Segmentation using Regions and Parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. (Cited on pages 23, 25, 85, and 91.)
- Hernán Badino, Uwe Franke, and David Pfeiffer. The Stixel World - A Compact Medium Level Representation of the 3D-World. In *DAGM Symposium*, 2009. (Cited on page 12.)
- Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. (Cited on page 24.)
- Leo Breiman. Random forest. *Machine Learning*, 45(5), 2001. (Cited on page 59.)
- Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and Recognition using Structure from Motion Point Clouds. In *European Conference on Computer Vision*, 2008. (Cited on pages 24, 32, 46, 57, and 58.)
- Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2), 2009. (Cited on pages 32, 34, and 42.)
- Wilhelm Burger. Zhang’s Camera Calibration Algorithm: In-Depth Tutorial and Implementation. Technical report, 2016. (Cited on page 57.)

- Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic Segmentation with Second-Order Pooling. In *European Conference on Computer Vision*, 2012. (Cited on pages 3, 4, 23, 25, 85, 91, 102, 103, and 104.)
- Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 131.)
- Marius Cordts, Timo Rehfeld, Markus Enzweiler, Uwe Franke, and Stefan Roth. Tree-Structured Models for Efficient Multi-Cue Scene Labeling. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017a. (Cited on pages 9, 113, 114, 122, 124, 125, and 131.)
- Marius Cordts, Timo Rehfeld, Lukas Schneider, David Pfeiffer, Markus Enzweiler, , Stefan Roth, Marc Pollefeys, and Uwe Franke. The Stixel World: A Medium-Level Representation of Traffic Scenes. In *Special Issue in Image and Vision Computing, Automotive Vision: Challenges, Trends, Technologies and Systems for Vision-Based Intelligent Vehicles*, 2017b. (Cited on pages 9, 14, 15, 19, 52, 64, and 129.)
- Nico Cornelis, Bastian Leibe, Kurt Cornelis, and Luc Van Gool. 3D Urban Scene Modeling Integrating Recognition and Reconstruction. *International Journal of Computer Vision*, 78(2-3), 2008. (Cited on page 33.)
- Arthur D Costea and Sergiu Nedevschi. Multi-Class Segmentation for Traffic Scenarios at Over 50 FPS. In *IEEE Intelligent Vehicles Symposium*, 2014. (Cited on page 23.)
- Camille Couprie, Clément Farabet, and Yann LeCun. Causal Graph-based Video Segmentation. In *IEEE International Conference on Image Processing*, 2013. (Cited on pages 25 and 96.)
- Gabriela Csurka, Diane Larlus, and Florent Perronnin. What is a good evaluation measure for semantic segmentation? In *British Machine Vision Conference*, 2013. (Cited on page 40.)
- Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual Categorization with Bags of Key-points. In *Workshop on Statistical Learning in Computer Vision (European Conference on Computer Vision)*, 2004. (Cited on page 20.)
- Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005. (Cited on pages xv and 3.)
- Roderick de Nijs, Sebastian Ramos, Gemma Roig, Xavier Boix, Luc Van Gool, and Kolja Kühnlenz. On-line Semantic Perception

- using Uncertainty. In *International Conference on Intelligent Robots and Systems*, 2012. (Cited on pages 25 and 96.)
- Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian Detection: An Evaluation of the State of the Art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4), 2012. (Cited on page 3.)
- Liam Ellis and Vasileios Zografos. Online Learning for Fast Segmentation of Moving Objects. In *Asian Conference on Computer Vision*, 2012. (Cited on pages 25 and 95.)
- Friedrich Erbs, Beate Schwarz, and Uwe Franke. Stixmentation - Probabilistic Stixel based Traffic Scene Labeling. In *British Machine Vision Conference*, 2012. (Cited on pages 13, 25, and 96.)
- Andreas Ess, Tobias Müller, Helmut Grabner, and Luc Van Gool. Segmentation-Based Urban Traffic Scene Understanding. In *British Machine Vision Conference*, 2009. (Cited on page 96.)
- Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 2010. (Cited on pages 39 and 40.)
- Clément Farabet, Camille Couprie, Laurent Najman, and Yann Lecun. Learning Hierarchical Features for Scene Labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 2013. (Cited on page 27.)
- Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2), 2004. (Cited on page 88.)
- Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), 2010. (Cited on page 3.)
- G. Floros and B. Leibe. Joint 2D-3D Temporally Consistent Semantic Segmentation of Street Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. (Cited on pages 25 and 96.)
- Uwe Franke, Clemens Rabe, Hernán Badino, and Stefan K. Gehrig. 6D-Vision: Fusion of Stereo and Motion for Robust Environment Perception. In *DAGM Symposium*, 2005. (Cited on pages 12, 95, and 96.)
- Uwe Franke, David Pfeiffer, Clemens Rabe, Carsten Knöppel, Markus Enzweiler, Fridtjof Stein, and Ralf G Herrtwich. Making Bertha See. In *Workshop on Computer Vision for Autonomous Driving (International Conference on Computer Vision)*, 2013. (Cited on page 36.)

- Friedrich Fraundorfer, Changchang Wu, and Marc Pollefeys. Combining Monocular and Stereo Cues for Mobile Robot Localization Using Visual Words. In *International Conference on Pattern Recognition*, 2010. (Cited on page 24.)
- Björn Fröhlich, Erik Rodner, and Joachim Denzler. Semantic Segmentation with Millions of Features: Integrating Multiple Cues in a Combined Random Forest Approach. In *Asian Conference on Computer Vision*, 2012. (Cited on pages 23 and 59.)
- Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *International Conference on Computer Vision*, 2009. (Cited on pages 3, 4, and 23.)
- Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 131.)
- Stefan K. Gehrig, Felix Eberli, and Thomas Meyer. A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching. In *International Conference on Computer Vision Systems*, 2009. (Cited on page 76.)
- Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. (Cited on pages 33 and 34.)
- Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1), 2006. (Cited on page 62.)
- Theo Gevers, Arjan Gijsenij, Joost van de Weijer, and Jan-Mark Geusebroek. Pixel-Based Photometric Invariance. In *Color in Computer Vision: Fundamentals and Applications*, chapter 4. John Wiley & Sons, 2012. (Cited on page 54.)
- Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. (Cited on page 4.)
- Stephen Gould. DARWIN: A Framework for Machine Learning and Computer Vision Research and Development. *Journal of Machine Learning Research*, 13, 2012. (Cited on pages 26, 79, 109, and 125.)
- Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *International Conference on Computer Vision*, 2009. (Cited on page 26.)

- Kristen Grauman and Trevor Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *International Conference on Computer Vision*, 2005. (Cited on pages 22 and 104.)
- Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient Hierarchical Graph-based Video Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. (Cited on page 24.)
- Saurabh Gupta, Pablo Arbeláez, and Jitendra Malik. Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. (Cited on pages 24 and 93.)
- Christian Häne, Christopher Zach, Andrea Cohen, Roland Angst, and Marc Pollefeys. Joint 3D Scene Reconstruction and Class Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. (Cited on page 6.)
- Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004. ISBN 0521540518. (Cited on page 33.)
- Hu He and Ben Upcroft. Nonparametric Semantic Segmentation for 3D Street Scenes. In *International Conference on Intelligent Robots and Systems*, 2013. (Cited on page 34.)
- Antonio Hernández-Vela, Miguel Ángel Bautista, Xavier Perez-Sala, and Victor Ponce. BoVDW: Bag-of-Visual-and-Depth-Words for Gesture Recognition. In *International Conference on Pattern Recognition*, 2012. (Cited on page 24.)
- Heiko Hirschmüller. Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2), 2008. (Cited on pages xv, 12, and 38.)
- Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8), 1998. (Cited on page 60.)
- Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering Surface Layout from an Image. *International Journal of Computer Vision*, 75(1), 2007. (Cited on page 24.)
- Rui Hu, Diane Larlus, and Gabriela Csurka. On the use of regions for semantic image segmentation. In *Indian Conference on Computer Vision, Graphics and Image Processing*, 2012. (Cited on page 85.)

- Christoph G. Keller, Markus Enzweiler, Marcus Rohrbach, David Fernández Llorca, Christoph Schnörr, and Darius M. Gavrilă. The Benefits of Dense Stereo for Pedestrian Detection. *IEEE Transactions on Intelligent Transportation Systems*, 12(4), 2011. (Cited on pages 3, 91, and 92.)
- Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009. ISBN 0262013192. (Cited on page 99.)
- John Krauskopf, David R. Williams, and David W. Heeley. Cardinal directions of color space. *Vision Research*, 22(9), 1982. (Cited on page 55.)
- Abhijit Kundu, Yin Li, Frank Daellert, Fuxin Li, and James M. Rehg. Joint Semantic Segmentation and 3D Reconstruction from Monocular Video. In *European Conference on Computer Vision*, 2014. (Cited on page 6.)
- Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature Space Optimization for Semantic Video Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 24.)
- Raphael Labayrade, Didier Aubert, and Jean-Philippe Tarel. Real Time Obstacle Detection in Stereovision on Non Flat Road Geometry Through V-disparity Representation. In *IEEE Intelligent Vehicles Symposium*, 2002. (Cited on page 58.)
- Lubor Ladický, Paul Sturgess, Chris Russell, Sunando Sengupta, Yalin Bastanlar, William Clocksin, and Philip H. S. Torr. Joint Optimisation for Object Class Segmentation and Dense Stereo Reconstruction. In *British Machine Vision Conference*, 2010. (Cited on pages xiv, 3, 6, 24, 26, 33, 34, 79, 109, 125, and 128.)
- Lubor Ladický, Chris Russell, Pushmeet Kohli, and Philip H. S. Torr. Associative Hierarchical Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. (Cited on page 88.)
- Lubor Ladický, Jianbo Shi, and Marc Pollefeys. Pulling Things out of Perspective. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. (Cited on page 34.)
- Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006. (Cited on pages 22 and 92.)
- Thomas Leung and Jitendra Malik. Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons. *International Journal of Computer Vision*, 43(1), 2001. (Cited on page 56.)

- David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *European Conference on Machine Learning*, 1998. (Cited on page 20.)
- Ming-Yu Liu, Shuoxin Lin, Srikumar Ramalingam, and Oncel Tuzel. Layered Interpretation of Street View Images. In *Robotics: Science and Systems*, 2015. (Cited on pages 27, 28, 79, 80, 108, 109, 124, 125, and 128.)
- Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on pages xv, 24, and 132.)
- David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2), 2004. (Cited on pages xv, 24, 90, and 101.)
- Rudolf Mester, Christian Conrad, and Alvaro Guevara. Multichannel Segmentation Using Contour Relaxation: Fast Super-Pixels and Temporal Propagation. In *Scandinavian Conference on Image Analysis*, 2011. (Cited on page 25.)
- Branislav Mičušík. Semantic segmentation of street scenes by super-pixel co-occurrence and 3d geometry. In *International Conference on Computer Vision*, 2009. (Cited on pages 23 and 24.)
- Ondrej Miksik, Daniel Munoz, J Andrew Bagnell, and Martial Hebert. Efficient Temporal Consistency for Streaming Video Scene Analysis. In *IEEE International Conference on Robotics and Automation*, 2013. (Cited on pages 25 and 96.)
- Frank Moosmann, Eric Nowak, and Frederic Jurie. Randomized clustering forests for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9), 2008. (Cited on pages xv, 21, 28, 29, 62, 89, 90, 104, and 117.)
- Andreas C. Müller and Sven Behnke. Learning Depth-Sensitive Conditional Random Fields for Semantic Segmentation of RGB-D Images. In *IEEE International Conference on Robotics and Automation*, 2014. (Cited on page 59.)
- Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018029. (Cited on page 97.)
- D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006. (Cited on page 21.)
- Peter Ochs and Thomas Brox. Object Segmentation in Video: A Hierarchical Variational Approach for Turning Point Trajectories into

- Dense Regions. In *International Conference on Computer Vision*, 2011. (Cited on page 25.)
- David Pfeiffer. *The Stixel World - A Compact Medium-level Representation for Efficiently Modeling Dynamic Three-dimensional Environments*. PhD thesis, Humboldt-Universität zu Berlin, 2011. (Cited on pages iii, iv, 7, 12, 14, 15, 52, 69, 75, 77, 78, and 79.)
- David Pfeiffer and Uwe Franke. Towards a Global Optimal Multi-Layer Stixel Representation of Dense 3D Data. In *British Machine Vision Conference*, 2011a. (Cited on page 105.)
- David Pfeiffer and Uwe Franke. Modeling Dynamic 3D Environments by Means of The Stixel World. *IEEE Intelligent Transportation Systems Magazine*, 3(3), 2011b. (Cited on page 13.)
- David Pfeiffer, Friedrich Erbs, and Uwe Franke. Pixels, stixels, and objects. In *Workshop on Computer Vision in Vehicle Technology (European Conference on Computer Vision)*, 2012. (Cited on page 13.)
- David Pfeiffer, Stefan K. Gehrig, and Nicolai Schneider. Exploiting the Power of Stereo Confidences. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. (Cited on page 19.)
- Sivalogeswaran Ratnasingam and Steve Collins. Study of the photodetector characteristics of a camera for color constancy in natural scenes. *Journal of the Optical Society of America*, 27(2), 2010. (Cited on page 55.)
- Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 4.)
- Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for Data: Ground Truth from Computer Games. In *European Conference on Computer Vision*, 2016. (Cited on page 131.)
- German Ros, Sebastian Ramos, Manuel Granados, Amir Bakhtiary, David Vazques, and Antonio M. Lopez. Vision-based Offline-Online Perception Paradigm for Autonomous Driving. In *IEEE Winter Conference on Applications of Computer Vision*, 2015. (Cited on page 34.)
- German Ros, Laura Sellart, Joanna Materzynska, David Vazques, and Antonio M. Lopez. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 131.)

- Timo Scharwächter and Uwe Franke. Low-Level Fusion of Color, Texture and Depth for Robust Road Scene Understanding. In *IEEE Intelligent Vehicles Symposium*, 2015. (Cited on pages 9 and 52.)
- Timo Scharwächter, Markus Enzweiler, Uwe Franke, and Stefan Roth. Efficient Multi-Cue Scene Segmentation. In *German Conference on Pattern Recognition*, 2013. (Cited on pages 9, 41, 84, 104, 105, and 106.)
- Timo Scharwächter, Markus Enzweiler, Uwe Franke, and Stefan Roth. Stixmantics: A medium-level model for real-time semantic scene understanding. In *European Conference on Computer Vision*, 2014. (Cited on pages 9, 84, 94, 105, and 106.)
- Sunando Sengupta, Eric Greveson, Ali Shahrokni, and Philip H. S. Torr. Urban 3D semantic modelling using stereo vision. In *IEEE International Conference on Robotics and Automation*, 2013. (Cited on page 34.)
- Abhishek Sharma, Oncel Tuzel, and David W. Jacobs. Deep Hierarchical Parsing for Semantic Segmentation. 2015. (Cited on pages 27, 28, 79, 80, 109, and 125.)
- Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-Class Object Recognition and Segmentation. In *European Conference on Computer Vision*, 2006. (Cited on pages 29 and 114.)
- Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic Texton Forests for Image Categorization and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008. (Cited on pages xiv, xvi, 23, 28, 29, 59, 109, 114, 115, 117, 125, and 130.)
- Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context. *International Journal of Computer Vision*, 81(1), 2009. (Cited on pages 3, 4, and 23.)
- Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for Computer Graphics: A Primer. *IEEE Computer Graphics and Applications*, 15, 1995. (Cited on pages 100 and 108.)
- Paul Sturgess, Karteek Alahari, Lubor Ladický, and Philip H. S. Torr. Combining Appearance and Structure from Motion Features for Road Scene Understanding. In *British Machine Vision Conference*, 2009. (Cited on pages 24, 32, 91, and 99.)
- Kevin Tang, Rahul Sukthankar, Jay Yagnik, and Li Fei-Fei. Discriminative Segment Annotation in Weakly Labeled Video. In *IEEE*

- Conference on Computer Vision and Pattern Recognition*, 2013. (Cited on page 25.)
- Joseph Tighe and Svetlana Lazebnik. SuperParsing: Scalable Non-parametric Image Parsing with Superpixels. In *European Conference on Computer Vision*, 2010. (Cited on page 25.)
- Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. Technical Report CMU-CS-91-132, Carnegie Mellon University, 1991. (Cited on page 96.)
- Zhuowen Tu and Xiang Bai. Auto-context and Its Application to High-level Vision Tasks and 3D Brain Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10), 2009. (Cited on page 23.)
- Ben Upcroft, Colin McManus, Winston Churchill, Will Maddern, and Paul M. Newman. Lighting Invariant Urban Street Classification. In *IEEE International Conference on Robotics and Automation*, 2014. (Cited on pages 54 and 55.)
- Amelio Vazquez-Reina, Shai Avidan, Hanspeter Pfister, and Eric Miller. Multiple Hypothesis Video Segmentation from Superpixel Flows. In *European Conference on Computer Vision*, 2010. (Cited on page 25.)
- Andrea Vedaldi and Brian Fulkerson. VLFeat: An Open and Portable Library of Computer Vision Algorithms. <http://www.vlfeat.org/>, 2008. (Cited on page 100.)
- Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Graph cut based image segmentation with connectivity priors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008. (Cited on page 88.)
- Remi Vieux, Jenny Benois-Pineau, Jean-Philippe Domenger, and Achille Braquelaire. Segmentation-based Multi-Class Semantic Object Detection. *Multimedia Tools and Applications*, 2011. (Cited on page 85.)
- Christian Wojek and Bernt Schiele. A Dynamic Conditional Random Field Model for Joint Labeling of Object and Scene Classes. In *European Conference on Computer Vision*, 2008. (Cited on pages 4, 25, and 96.)
- Christian Wojek, Stefan Walk, Stefan Roth, Konrad Schindler, and Bernt Schiele. Monocular Visual Scene Understanding: Understanding Multi-Object Traffic Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4), 2013. (Cited on pages 25 and 96.)

- Jianxin Wu. A Fast Dual Method for HIK SVM Learning. In *European Conference on Computer Vision*, 2010. (Cited on pages [xv](#) and [93](#).)
- Philippe Xu, Franck Davoine, Jean-Baptiste Bordes, Huijing Zhao, and Thierry Denoeux. Information Fusion on Oversegmented Images: An Application for Urban Scene Understanding. In *IAPR International Conference on Machine Vision Applications*, 2013. (Cited on page [34](#).)
- Chenxi Zhang, Liang Wang, and Ruigang Yang. Semantic Segmentation of Urban Scenes Using Dense Depth Maps. In *European Conference on Computer Vision*, 2010. (Cited on pages [23](#), [24](#), and [57](#).)
- J. Zhang, M. Marszalek, Svetlana Lazebnik, and Cordelia Schmid. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *International Journal of Computer Vision*, 73(2), 2006. (Cited on page [4](#).)
- Julius Ziegler, Thao Dang, Uwe Franke, Henning Lategahn, Philipp Bender, Markus Schreiber, Tobias Strauss, Nils Appenrodt, Christoph G Keller, Eberhard Kaus, Christoph Stiller, Ralf G Hertrich, Clemens Rabe, David Pfeiffer, Frank Lindner, Fridtjof Stein, Friedrich Erbs, MarkusENZweiler, Carsten Knöppel, Jochen Hipp, Martin Haueis, Maximilian Trepte, Carsten Brenk, Andreas Tamke, Muhammad Ghanaat, Markus Braun, Armin Joos, Hans Fritz, Horst Mock, Martin Hein, and Eberhard Zeeb. Making Bertha Drive: An Autonomous Journey on a Historic Route. *IEEE Intelligent Transportation Systems Magazine*, 6(2), 2014. (Cited on page [36](#).)

COLOPHON

This document was typeset using the typographical look-and-feel classicthesis developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". classicthesis is available for both L^AT_EX and L^YX:

<http://code.google.com/p/classicthesis/>

Happy users of classicthesis usually send a real postcard to the author, a collection of postcards received so far is featured here:

<http://postcards.miede.de/>

Final Version as of March 10, 2018